Guía para desarrolladores

AWS SDK for Ruby

AWS SDK for Ruby: Guía para desarrolladores

Table of Contents

¿Qué es AWS SDK para Ruby?

Le damos la bienvenida a la guía para desarrolladores de AWS SDK para Ruby. AWS SDK para Ruby proporciona bibliotecas de soporte para casi todos los Servicios de AWS, incluidos Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2) y Amazon DynamoDB.

La Guía para desarrolladores de AWS SDK para Ruby proporciona información sobre cómo instalar, configurar y usar AWS SDK para Ruby para crear aplicaciones Ruby que usen Servicios de AWS.

Introducción a AWS SDK para Ruby

Documentación y recursos adicionales

Más recursos para desarrolladores de AWS SDK para Ruby; se encuentran en:

- Guía de referencia de las herramientas y los SDK de AWS: incluye configuraciones, funciones y otros conceptos básicos comunes a los SDK de AWS.
- Referencia de la API de AWS SDK for Ruby versión 3
- Repositorio de ejemplos de código AWS en GitHub
- RubyGems.org: la última versión del SDK está modularizada en gemas específicas de cada servicio, disponibles aquí
  - Servicios compatibles: enumera todas las gemas compatibles con AWS SDK para Ruby
- AWSFuente de SDK para Ruby en GitHub:
  - Fuente y README
  - Registros de cambios bajo cada gema
  - Pasar de v2 a v3
  - Problemas
  - Notas sobre la actualización principal
- Blog de desarrolladores
- Canal de Gitter
- @awsforruby en Twitter

Implementación en la nube de AWS

Puede utilizar los Servicios de AWS, por ejemplo, AWS Elastic Beanstalk, AWS OpsWorks y AWS CodeDeploy para implementar su aplicación en la nube de AWS. Para implementar aplicaciones Ruby con Elastic Beanstalk, consulte Implementación de aplicaciones de Elastic Beanstalk en Using con la CLI de EB y Git en la Guía del desarrollador de AWS Elastic Beanstalk. Para implementar una aplicación de Ruby on Rails con AWS OpsWorks, consulte Implementación de aplicaciones Ruby on Rails en AWS OpsWorks. Para obtener información general sobre los servicios de implementación de AWS, consulte Información general de las opciones de implementación en AWAWS.

Mantenimiento y compatibilidad de las versiones principales del SDK

Para obtener información sobre el mantenimiento y la compatibilidad con las principales versiones del SDK y sus dependencias subyacentes, consulte lo siguiente en la Guía de Referencia de SDK y herramientas de AWS:

- Política de mantenimiento de SDK y herramientas AWS
- Matriz de compatibilidad para versiones de SDK y herramientas AWS

Introducción a AWS SDK para Ruby

Aprenda a instalar, configurar y usar el SDK para crear una aplicación Ruby para acceder a un recurso de AWS mediante programación.

Temas

Autenticación de SDK con AWS

Debe establecer cómo se autentica el código con AWS cuando se desarrolla con Servicios de AWS. Puede configurar el acceso a los recursos de AWS mediante programación de diferentes maneras, según el entorno y el acceso a AWS disponibles.

Para elegir el método de autenticación y configurarlo para el SDK, consulte Autenticación y acceso en la Guía de referencia de las herramientas y los SDK de AWS.

Recomendamos que los nuevos usuarios que desarrollen a nivel local y no dispongan de un método de autenticación por parte de su empresa configuren AWS IAM Identity Center. Este método incluye la instalación de AWS CLI para facilitar la configuración y para iniciar sesión con regularidad en el portal de acceso de AWS. Si elige este método, su entorno debería contener los siguientes elementos después de completar el procedimiento de autenticación del IAM Identity Center descrito en la Guía de referencia de las herramientas y los SDK de AWS:

- La AWS CLI, que se utiliza para iniciar una sesión en el portal de acceso a AWS antes de ejecutar la aplicación.
- Un archivo `config` compartido de AWS que tiene un perfil `[default]` con un conjunto de valores de configuración a los que se puede hacer referencia desde el SDK. Para encontrar la ubicación de este archivo, consulte Ubicación de los archivos compartidos en la Guía de referencia de herramientas y AWS SDK.
- El archivo `config` compartido establece la configuración de `region`. Esto establece la Región de AWS predeterminada que el SDK usa para las solicitudes de AWS. Esta región se usa para las solicitudes de servicio del SDK que no tienen especificadas una región.

- El SDK utiliza la configuración de proveedor de token de SSO del perfil para adquirir las credenciales antes de enviar las solicitudes a AWS. El valor `sso_role_name`, que es un rol de IAM conectado a un conjunto de permisos del Centro de identidades de IAM, permite el acceso a los Servicios de AWS utilizados en la aplicación.

  El siguiente archivo `config` de ejemplo muestra la configuración de un perfil predeterminado con el proveedor de token de SSO. La configuración `sso_session` del perfil hace referencia a la sección llamada `sso-session`. La sección `sso-session` contiene la configuración para iniciar una sesión en el portal de acceso a AWS.

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

AWS SDK para Ruby no necesita añadir paquetes adicionales (por ejemplo, `SSO` y `SSOOIDC`) a la aplicación para utilizar la autenticación del IAM Identity Center.

Iniciar una sesión en el portal de acceso a AWS

Antes de ejecutar una aplicación que accede a Servicios de AWS, necesita una sesión activa en el portal de acceso a AWS para que el SDK utilice la autenticación del IAM Identity Center para resolver las credenciales. En función de la duración de las sesiones configuradas, el acceso terminará por caducar y el SDK detectará un error de autenticación. Para iniciar sesión en el portal de acceso a AWS, ejecute el siguiente comando en la AWS CLI.

```
aws sso login
```

Si sigue la guía y utiliza una configuración de perfil predeterminada, no necesita llamar al comando con una opción `--profile`. Si la configuración del proveedor de token de SSO utiliza un perfil con nombre, el comando es `aws sso login --profile named-profile`.

Para comprobar si ya tiene una sesión activa, ejecute el siguiente comando de AWS CLI.

```
aws sts get-caller-identity
```

Si su sesión está activa, la respuesta a este comando debe indicar la cuenta y el conjunto de permisos del IAM Identity Center configurados en el archivo `config` compartido.

> **Note**
>
> Si ya tiene una sesión activa en el portal de acceso a AWS y ejecuta `aws sso login`, no tendrá que proporcionar credenciales.
> Es posible que el proceso de inicio de sesión le permita el acceso de la AWS CLI a los datos. Como AWS CLI se ha creado con el SDK para Python, los mensajes de permiso pueden contener variaciones del nombre `botocore`.

Información adicional de autenticación

Los usuarios humanos, que también reciben el nombre de identidades humanas, son las personas, los administradores, los desarrolladores, los operadores y los consumidores de las aplicaciones. Deben tener una identidad para acceder a los entornos y aplicaciones de AWS. Los usuarios humanos que son miembros de su organización (es decir, usted, el desarrollador) se conocen como identidades de personal.

Utilice credenciales temporales al acceder a AWS. Puede utilizar un proveedor de identidades para los usuarios humanos para proporcionar acceso federado a las cuentas de AWS asumiendo roles, que proporcionan credenciales temporales. Si desea administrar el acceso de manera centralizada, se recomienda utilizar AWS IAM Identity Center (IAM Identity Center) para administrar el acceso a las cuentas y los permisos de esas cuentas. Para obtener más alternativas, consulte lo siguiente:

- Para obtener más información sobre las prácticas recomendadas, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.
- Para crear credenciales de AWS de corta duración, consulte Credenciales de seguridad temporales en la Guía del usuario de IAM.
- Para obtener más información sobre otros proveedores de credenciales de AWS SDK para Ruby, consulte los proveedores de credenciales estandarizados en la Guía de referencia de las herramientas y los SDK de AWS.

Instalar AWS SDK para Ruby

Esta sección incluye los requisitos previos y las instrucciones de instalación de AWS SDK para Ruby.

Requisitos previos

Antes de utilizar AWS SDK para Ruby debe autenticarse con AWS. Para obtener información acerca de la configuración de la autenticación, consulte Autenticación de SDK con AWS.

Instalación del SDK

Puede instalar AWS SDK para Ruby como lo haría con cualquier gema de Ruby. Las gemas están disponibles en RubyGems. AWS SDK para Ruby está diseñado para ser modular y está separado por Servicio de AWS. La instalación de toda la gema `aws-sdk` es larga y puede llevar más de una hora.

Le recomendamos que instale solo las gemas para los Servicios de AWS que utilice. Su denominación sigue el modelo `aws-sdk-service_abbreviation` y la lista completa se encuentra en la tabla Servicios compatibles del archivo README de AWS SDK para Ruby. Por ejemplo, la gema para interactuar con el servicio Amazon S3 está disponible directamente en `aws-sdk-s3`.

Administrador de versiones de Ruby

En lugar de usar el Ruby del sistema, recomendamos usar un administrador de versiones de Ruby como el siguiente:

- RVM
- chruby
- rbenv

Por ejemplo, si utiliza un sistema operativo Amazon Linux 2, puede usar los siguientes comandos para actualizar RVM, enumerar las versiones de Ruby disponibles y, a continuación, elegir la versión que desea usar para el desarrollo con AWS SDK para Ruby. La versión mínima de Ruby requerida es 2.3.

```
$ rvm get head
$ rvm list known
$ rvm install ruby-3.1.3
```

```
$ rvm --default use 3.1.3
```

Bundler

Si utiliza Bundler, los siguientes comandos instalan la gema AWS SDK para Ruby para Amazon S3:

1. Instale Bundler y cree el archivo `Gemfile`:

   ```
   $ gem install bundler
   $ bundle init
   ```

2. Abra el archivo `Gemfile` que ha creado y añada una línea de `gem` para cada gema de servicio de AWS que utilice su código. Para seguir con el ejemplo de Amazon S3, agregue la siguiente línea de texto al final del archivo:

   ```
   gem "aws-sdk-s3"
   ```

3. Guarde el archivo Gemfile.
4. Instale las dependencias especificadas en su archivo `Gemfile`:

   ```
   $ bundle install
   ```

Tutorial de Hello World para AWS SDK para Ruby

Descubra Amazon S3 utilizando AWS SDK para Ruby. En el siguiente ejemplo se muestra una lista de sus buckets de Amazon S3.

Escriba el código

Copie y pegue el siguiente código en un nuevo archivo de origen. Nombre el archivo `hello-s3.rb`.

```
require "aws-sdk-s3"

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
```

```
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts "Found these buckets:"
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS SDK para Ruby está diseñado para ser modular y está separado por Servicio de AWS. Una vez instalada la gema, la instrucción de `require` que aparece en la parte superior del archivo fuente de Ruby importa las clases y los métodos del SDK de AWS para el servicio Amazon S3. Para obtener una lista completa de las gemas de los servicios de AWS disponibles, consulte la tabla Servicios compatibles del archivo README de AWS SDK para Ruby.

```
require 'aws-sdk-s3'
```

Ejecución del programa

Abra un comando para ejecutar su programa Ruby. La sintaxis de comando habitual para ejecutar un programa Ruby es:

```
ruby [source filename] [arguments...]
```

Este ejemplo de código no utiliza argumentos. Para ejecutar este código, introduzca el siguiente comando del sistema:

```
$ ruby hello-s3.rb
```

Nota para usuarios de Windows

Cuando utiliza certificados SSL en Windows y ejecuta el código Ruby, podría ver un error similar al siguiente.

```
C:\Ruby>ruby buckets.rb
C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `connect': SSL_connect returned=1
 errno=0 state=SSLv3 read server certificate B: certificate verify failed
 (Seahorse::Client::NetworkingError)
        from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `block in connect'

        from C:/Ruby200-x64/lib/ruby/2.0.0/timeout.rb:66:in `timeout'
        from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `connect'
        from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:862:in `do_start'
        from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:857:in `start'
...
```

Para solucionar este problema, añada la siguiente línea a su archivo de código de Ruby, en algún lugar antes de la primera llamada a AWS.

```
Aws.use_bundled_cert!
```

Tenga en cuenta que si solo utiliza la gema `aws-sdk-s3` en su programa de Ruby, también tendrá que añadir la gema `aws-sdk-core` para utilizar el certificado agrupado.

Siguientes pasos

Para probar muchas otras operaciones de Amazon S3, consulte el repositorio de ejemplos de AWS código en GitHub.

Uso de AWS Cloud9 con AWS SDK para Ruby

AWS Cloud9 es un entorno de desarrollo integrado (IDE) basado en la web que contiene una colección de herramientas que se utilizan para escribir código, así como compilar, ejecutar, probar,

depurar y publicar software en la nube. Puede utilizar AWS Cloud9 con AWS SDK para Ruby para escribir y ejecutar el código Ruby con tan solo un navegador. AWS Cloud9 incluye herramientas, como un editor de código y un terminal. Dado que el IDE de AWS Cloud9 está basado en la nube, puede trabajar en sus proyectos desde la oficina, desde su casa o desde cualquier lugar con un equipo conectado a Internet. Para obtener información general sobre AWS Cloud9, consulte la Guía del usuario de AWS Cloud9.

Siga estas instrucciones para configurar AWS Cloud9 con AWS SDK para Ruby:

- Paso 1: Configurar su Cuenta de AWS para utilizar AWS Cloud9
- Paso 2: Configurar su entorno de desarrollo de AWS Cloud9
- Paso 3: Configurar AWS SDK para Ruby
- Paso 4: Descargar el código de ejemplo
- Paso 5: Ejecutar el código de ejemplo

Paso 1: Configurar su Cuenta de AWS para utilizar AWS Cloud9

Para usar AWS Cloud9, inicie sesión en la consola de AWS Cloud9 desde AWS Management Console.

> **Note**
>
> Si utiliza AWS IAM Identity Center para autenticarse, es posible que necesite añadir el permiso necesario de `iam:ListInstanceProfilesForRole` a la política asociada al usuario en la consola de IAM.

Para configurar una entidad de AWS en su cuenta de AWS para acceder a AWS Cloud9 e iniciar sesión en la consola de AWS Cloud9, consulte la sección sobre configuración de equipos para AWS Cloud9AWS Cloud9 en la AWS Cloud9Guía del usuario de AWS Cloud9.

Paso 2: Configurar su entorno de desarrollo de AWS Cloud9

Después de iniciar sesión en la consola de AWS Cloud9, utilice la consola para crear un entorno de desarrollo de AWS Cloud9. Una vez creado el entorno, AWS Cloud9 abrirá el IDE en dicho entorno.

Para obtener más información, consulte la sección Crear un entorno en AWS Cloud9 en la guía del usuario de AWS Cloud9.

> **Note**
>
> Al crear el entorno en la consola por primera vez, le recomendamos que elija la opción Create a new instance for environment (EC2) [Crear una nueva instancia para el entorno (EC2)]. Esta opción le indica a AWS Cloud9 que cree un entorno, lance una instancia de Amazon EC2 y, a continuación, conecte la nueva instancia al nuevo entorno. Esta es la forma más rápida de empezar a utilizar AWS Cloud9.

Si el terminal todavía no está abierto en el IDE, ábralo. En la barra de menú del IDE, elija Window, New Terminal (Ventana, Nuevo terminal). Puede utilizar la ventana de terminal para instalar herramientas y crear sus aplicaciones.

Paso 3: Configurar AWS SDK para Ruby

Después de que AWS Cloud9 abra el IDE para su entorno de desarrollo, utilice la ventana de terminal para configurar AWS SDK para Ruby en su entorno.

Puede instalar AWS SDK para Ruby como lo haría con cualquier gema de Ruby. Las gemas están disponibles en RubyGems. AWS SDK para Ruby está diseñado para ser modular y está separado por Servicio de AWS. La instalación de toda la gema `aws-sdk` es larga y puede llevar más de una hora.

Le recomendamos que instale solo las gemas para los Servicios de AWS que utilice. Su denominación sigue el modelo `aws-sdk-`*`service_abbreviation`* y la lista completa se encuentra en la tabla Servicios compatibles del archivo README de AWS SDK para Ruby. Por ejemplo, la gema para interactuar con el servicio Amazon S3 está disponible directamente en `aws-sdk-s3`.

Administrador de versiones de Ruby

En lugar de usar el Ruby del sistema, recomendamos un administrador de versiones de Ruby como el siguiente:

- RVM
- chruby
- rbenv

Por ejemplo, si utiliza el sistema operativo Amazon Linux 2, puede usar los siguientes comandos para actualizar RVM, enumerar las versiones de Ruby disponibles y, a continuación, elegir la versión que desea usar para el desarrollo con AWS SDK para Ruby. La versión mínima requerida es 2.3.

```
$ rvm get head
$ rvm list known
$ rvm install ruby-3.1.3
$ rvm --default use 3.1.3
```

Bundler

Si utiliza Bundler, los siguientes comandos instalan la gema AWS SDK para Ruby para Amazon S3:

1. Instale Bundler y cree el archivo `Gemfile`:

   ```
   $ gem install bundler
   $ bundle init
   ```

2. Abra el archivo `Gemfile` que ha creado y añada una línea de `gem` para cada gema de servicio de AWS que utilice su código. Para seguir con el ejemplo de Amazon S3, agregue la siguiente línea de texto al final del archivo:

   ```
   gem "aws-sdk-s3"
   ```

3. Guarde el archivo Gemfile.
4. Instale las dependencias especificadas en su `Gemfile`:

   ```
   $ bundle install
   ```

Paso 4: Descargar el código de ejemplo

Utilice la ventana de terminal para descargar código de ejemplo de AWS SDK para Ruby en el entorno de desarrollo de AWS Cloud9.

Para descargar una copia de todos los ejemplos de código que se utilizan en la documentación oficial del SDK de AWS en el directorio raíz del entorno, ejecute el siguiente comando:

```
$ git clone https://github.com/awsdocs/aws-doc-sdk-examples.git
```

Los ejemplos de código de AWS SDK para Ruby están disponibles en el directorio de `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby`, donde `ENVIRONMENT_NAME` es el nombre de su entorno de desarrollo.

Para seguir con un ejemplo de Amazon S3, le recomendamos empezar con un ejemplo de código de `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby/example_code/s3/bucket_list.rb`. Utilice la ventana de terminal para acceder al directorio de `s3` y enumerar los archivos.

```
$ cd aws-doc-sdk-examples/ruby/example_code/s3
$ ls
```

Para abrir el archivo en AWS Cloud9, puede hacer clic directamente en `bucket_list.rb` en la ventana de terminal.

Si necesita más ayuda para entender los ejemplos de código, consulte la sección Ejemplos de código de AWS SDK para Ruby.

Paso 5: Ejecutar el código de ejemplo

Para ejecutar código en el entorno de desarrollo de AWS Cloud9, seleccione el botón Ejecutar en la barra de menú superior. AWS Cloud9 detecta automáticamente la extensión del archivo `.rb` y utiliza el ejecutor de Ruby para ejecutar el código. Para obtener más información sobre cómo ejecutar código en AWS Cloud9, consulte la sección Ejecutar el código de la Guía del usuario de AWS Cloud9.

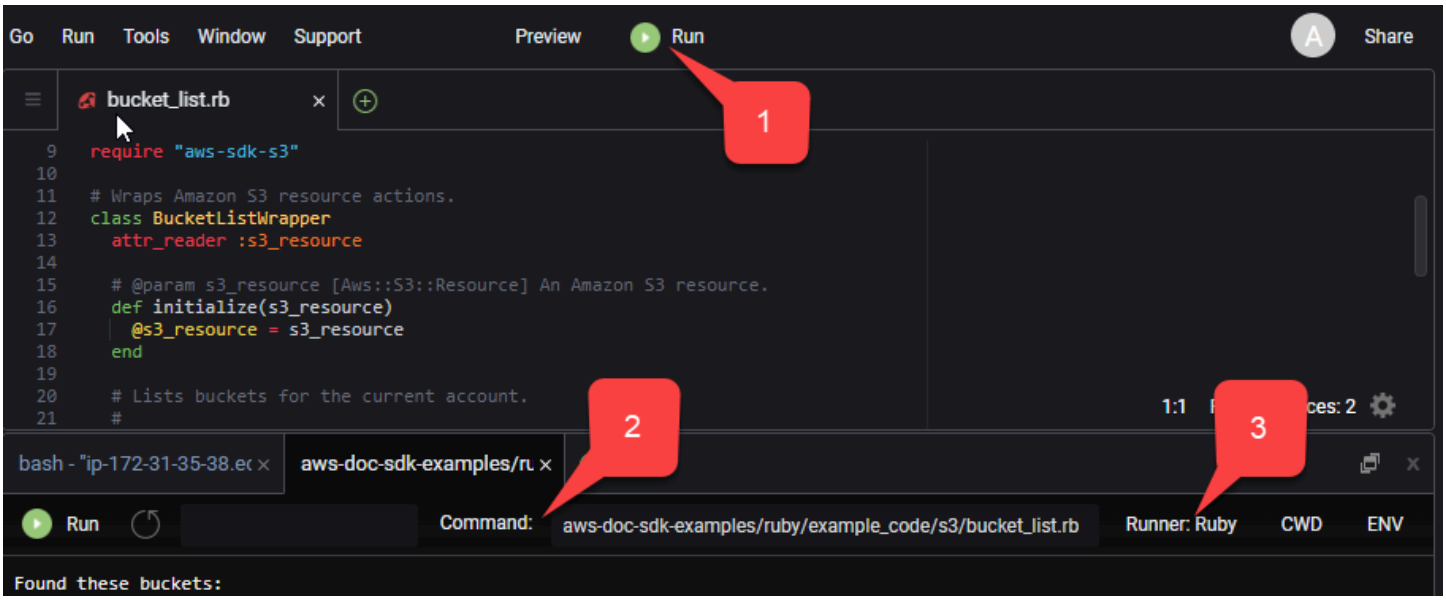
En la siguiente captura de pantalla, observe estas áreas básicas:

- 1: Ejecutar. El botón Ejecutar se encuentra en la barra de menú superior. Se abrirá una nueva pestaña para ver los resultados.

  > **Note**
  >
  > También puede crear nuevas configuraciones de ejecución manualmente. En la barra de menús, elija Ejecutar, Configuraciones de ejecución, Nueva configuración de ejecución.

- 2: Comando. AWS Cloud9 rellena el cuadro de texto Comando con la ruta y el nombre del archivo que está ejecutando. Si el código espera que se le pasen parámetros de línea de comandos, puede añadirlos a la línea de comandos del mismo modo que lo haría al ejecutar el código a través de una ventana de terminal.

- 3: Ejecutor. AWS Cloud9 detecta que la extensión del archivo es `.rb` y selecciona el ejecutor de Ruby para ejecutar su código.



En la pestaña se muestra cualquier resultado generado a partir del código en ejecución.

Configurar el AWS SDK para Ruby

Obtenga información sobre cómo configurar AWS SDK para Ruby. Debe establecer cómo se autentica el código con AWS cuando desarrolla con Servicios de AWS. También debe configurar la Región de AWS en la que que lo desea usar.

Cadena de proveedores de credenciales

Todos los SDK tienen una serie de lugares (o fuentes) que consultan para obtener credenciales válidas y utilizarlas para realizar una solicitud a un Servicio de AWS. Una vez que se encuentran las credenciales válidas, se detiene la búsqueda. A esta búsqueda sistemática, se le denomina cadena predeterminada de proveedores de credenciales.

Para cada paso de la cadena, hay diferentes maneras de establecer los valores. La configuración de los valores directamente en el código siempre tiene prioridad, seguida de la configuración como variables de entorno y, por último, en el archivo compartido `config` de AWS. Para obtener más información, consulte Prioridad de configuración en la Guía de referencia de las herramientas y los SDK de AWS.

La Guía de referencia de las herramientas y los SDK de AWS contiene información sobre los ajustes de configuración de SDK que utilizan todos los SDK de AWS y los AWS CLI. Para obtener más información sobre cómo configurar el SDK a través del archivo compartido `config` de AWS, consulte Archivos de configuración y credenciales compartidos. Para obtener más información sobre cómo configurar el SDK mediante la configuración de variables de entorno, consulte Compatibilidad con variables de entorno.

Para autenticarse en AWS, AWS SDK para Ruby comprueba los proveedores de credenciales en el orden que se indica en la siguiente tabla.

Proveedor de credenciales por prioridad	Guía de referencia de las herramientas y los SDK de AWS	Referencia de la API de AWS SDK for Ruby
Credenciales estáticas	Claves de acceso de AWS	`Aws::Credentials` `Aws::SharedCredentials`

Proveedor de credenciales por prioridad	Guía de referencia de las herramientas y los SDK de AWS	Referencia de la API de AWS SDK for Ruby
Token de identidad web de AWS Security Token Service (AWS STS)	Asumir el rol de proveedor de credenciales Uso de `role_arn`, `role_session_name` y `web_identity_token_file`	`Aws::AssumeRoleWebIdentityCredentials`
AWS IAM Identity Center. En esta guía, consulte Autenticación de SDK con AWS.	Proveedor de credenciales del IAM Identity Center	`Aws::SSOCredentials`
Proveedor de entidades de confianza (como `AWS_ROLE_ARN`). En esta guía, consulte Creación de un token de acceso de AWS STS.	Asumir el rol de proveedor de credenciales Uso de `role_arn` y `role_session_name`	`Aws::AssumeRoleCredentials`
Proveedor de credenciales de proceso	Proveedor de credenciales de proceso	`Aws::ProcessCredentials`
Credenciales de Amazon Elastic Container Service (Amazon ECS)	Proveedor de credenciales de contenedor	`Aws::ECSCredentials`
Credenciales de perfil de instancia de Amazon Elastic Compute Cloud (Amazon EC2) (proveedor de credencia les IMDS)	Proveedor de credenciales IMDS	`Aws::InstanceProfileCredentials`

Si se establece la variable de entorno de Ruby `AWS_SDK_CONFIG_OPT_OUT` de AWS SDK para Ruby, no se analizarán las credenciales del archivo compartido `config` de AWS, normalmente en `~/.aws/config`.

Si siguió el enfoque recomendado para los nuevos usuarios para empezar, configuró la autenticación AWS IAM Identity Center en Autenticación de SDK con AWS del tema Introducción. Otros métodos de autenticación son útiles en diferentes situaciones. Para evitar riesgos de seguridad, recomendamos utilizar siempre credenciales a corto plazo. Para conocer otros procedimientos de métodos de autenticación, consulte Autenticación y acceso en la Guía de referencia de las herramientas y los SDK de AWS.

Creación de un token de acceso de AWS STS

Para asumir un rol, se utiliza un conjunto de credenciales de seguridad temporales para acceder a los recursos de AWS a los que de otro modo usted no tendría acceso. Las credenciales temporales incluyen un ID de clave de acceso, una clave de acceso secreta y un token de seguridad. Puede usar el método `Aws::AssumeRoleCredentials` para crear un token de acceso AWS Security Token Service (AWS STS).

En el siguiente ejemplo se utiliza un token de acceso para crear un objeto de cliente de Amazon S3, donde `linked::account::arn` es el nombre de recurso de Amazon (ARN) del rol que se va a asumir, y `session-name` es un identificador de la sesión del rol asumido.

```
role_credentials = Aws::AssumeRoleCredentials.new(
  client: Aws::STS::Client.new,
  role_arn: "linked::account::arn",
  role_session_name: "session-name"
)

s3 = Aws::S3::Client.new(credentials: role_credentials)
```

Para obtener más información sobre la configuración de `role_arn` o `role_session_name`, o sobre cómo configurarlos utilizando el archivo compartido `config` de AWS, consulte Asumir el rol de proveedor de credenciales en la Guía de referencia de las herramientas y los SDK de AWS.

Configuración de una región

Debe configurar una región cuando se utiliza la mayoría de los Servicios de AWS. AWS SDK para Ruby busca una región en el orden que se indica a continuación:

1. Configuración de la región en un objeto de cliente o recurso
2. Configuración de la región mediante `Aws.config`

3. Configuración de la región mediante variables de entorno
4. Configuración de la región mediante el archivo compartido `config`

Para obtener más información sobre la configuración de `region`, consulte Región de AWS en la Guía de referencia de las herramientas y los SDK de AWS. En el resto de esta sección se describe cómo configurar una región, comenzando por el enfoque más común.

Configuración de la región mediante el archivo compartido **`config`**

Defina la región configurando la variable `region` en el archivo compartido `config` de AWS. Para obtener más información sobre el archivo compartido `config`, consulte Archivos de configuración y credenciales compartidos en la Guía de referencia de las herrramientas y los SDK de AWS.

Ejemplo de configuración de este valor en el archivo `config`:

```
[default]
region = us-west-2
```

El archivo compartido `config` no se comprueba si la variable de entorno `AWS_SDK_CONFIG_OPT_OUT` está configurada.

Configuración de la región mediante variables de entorno

Configure la región mediante estableciendo la variable de entorno `AWS_REGION`.

Use el comando `export` para establecer esta variable en sistemas basados en Unix, como Linux o macOS. En el siguiente ejemplo se establece la región en `us-west-2`.

```
export AWS_REGION=us-west-2
```

Para establecer esta variable en Windows, utilice el comando `set`. En el siguiente ejemplo se establece la región en `us-west-2`.

```
set AWS_REGION=us-west-2
```

Configuración de la región con **`Aws.config`**

Configure la región añadiendo un valor `region` en el hash `Aws.config`. En el siguiente ejemplo se actualiza el hash `Aws.config` para utilizar la región `us-west-1`.

```
Aws.config.update({region: 'us-west-1'})
```

Todos los clientes o recursos que cree posteriormente estarán asociados a esta región.

Configuración de la región en un objeto de recurso

Configure la región al crear un cliente o recurso de AWS. En el siguiente ejemplo se crea un objeto de recurso de Amazon S3; en la región `us-west-1`. Elija la región correcta para sus recursos de AWS. Un objeto de cliente de servicio es inmutable, por lo que debe crear un cliente nuevo para cada servicio al que realice solicitudes y para realizar solicitudes al mismo servicio con una configuración diferente.

```
s3 = Aws::S3::Resource.new(region: 'us-west-1')
```

Configuración de un punto de conexión no estándar

La región se utiliza para crear un punto de conexión SSL que se utilizará en las solicitudes de AWS. Si necesita utilizar un punto de conexión no estándar en la región que ha seleccionado, añada una entrada de `endpoint` para `Aws.config`. Como alternativa, configure el `endpoint:` al crear un cliente de servicio o un objeto de recurso. En el siguiente ejemplo se crea un objeto de recurso de Amazon S3 en el punto de conexión `other_endpoint`.

```
s3 = Aws::S3::Resource.new(endpoint: other_endpoint)
```

Para utilizar un punto de conexión de su elección para las solicitudes de la API y para hacer que esa opción persista, consulte la opción de configuración de puntos de conexión específicos del servicio de la Guía de referencia de las herramientas y los SDK de AWS.

Usar el AWS SDK para Ruby

Esta sección contiene información sobre el desarrollo de software con AWS SDK para Ruby; por ejemplo, acerca de cómo utilizar algunas de las características avanzadas del SDK.

La Guía de referencia de las herramientas y los SDK de AWS también contiene configuraciones, características y otros conceptos fundamentales comunes a muchos de los SDK de AWS.

Temas

Usar la utilidad REPL de AWS SDK para Ruby

La gema `aws-sdk` incorpora una interfaz de línea de comandos interactiva Read-Eval-Print-Loop (REPL) en la que puede probar SDK para Ruby y ver los resultados de inmediato. Las gemas de SDK para Ruby están disponibles en RubyGems.org.

Requisitos previos

- Instalar AWS SDK para Ruby.
- El `aws-v3.rb` está ubicado en la gema `aws-sdk-resources`. La gema `aws-sdk-resources` también está incluida en la gema `aws-sdk` principal.
- Necesitará una biblioteca xml, como la gema `rexml`.
- Si bien el programa funciona con Interactive Ruby Shell (`irb`), recomendamos instalar la gema `pry` porque proporciona un entorno REPL más potente.

Configuración de Bundler

Si utiliza Bundler, las siguientes actualizaciones de su archivo `Gemfile` abordarán las gemas necesarias:

1. Abra el archivo `Gemfile` que creó al instalar el AWS SDK para Ruby. Añada las líneas siguientes al archivo:

```
gem "aws-sdk"
gem "rexml"
gem "pry"
```

2. Guarde el archivo Gemfile.
3. Instale las dependencias especificadas en su archivo `Gemfile`:

```
$ bundle install
```

Ejecución de REPL

Puede obtener acceso a REPL ejecutando `aws-v3.rb` en la línea de comandos.

```
aws-v3.rb
```

Si lo prefiere, puede habilitar el registro de red HTTP mediante la configuración del indicador detallado. El registro de red HTTP proporciona información sobre la comunicación entre el AWS SDK para Ruby y AWS. Tenga en cuenta que el indicador detallado también añade una sobrecarga que puede enlentecer la ejecución del código.

```
aws-v3.rb -v
```

SDK para Ruby incluye clases de cliente que proporcionan interfaces para los Servicios de AWS. Cada clase de cliente admite un Servicio de AWS determinado. En la utilidad REPL, cada clase de servicio tiene un auxiliar que devuelve un nuevo objeto de cliente para interactuar con ese servicio. El nombre del auxiliar será el nombre del servicio convertido a minúsculas. Por ejemplo, los nombres de los objetos auxiliares de Amazon S3 y Amazon EC2 son `s3` y `ec2`, respectivamente. Para enumerar los buckets de Amazon S3 de su cuenta, puede introducir `s3.list_buckets` en el cuadro de diálogo.

Puede escribir `quit` en el mensaje de REPL para salir.

Usar el SDK con Ruby on Rails

Ruby on Rails proporciona un marco de desarrollo web que facilita la creación de sitios web con Ruby.

AWS proporciona la gema `aws-sdk-rails` para que la integración con Rails sea sencilla. Puede utilizar AWS Elastic Beanstalk, AWS OpsWorks, AWS CodeDeploy o AWS Rails Provisioner para implementar y ejecutar aplicaciones de Rails en la nube de AWS.

Para obtener información sobre cómo instalar y usar la gema `aws-sdk-rails`, consulte el repositorio de GitHub https://github.com/aws/aws-sdk-rails.

Consejo de depuración: Obtener información del rastreo de red de un cliente

Puede obtener información del rastro de red de un cliente de AWS configurando el `http_wire_trace` booleano. La información del rastro de red ayuda a diferenciar los cambios de cliente, los problemas de servicio y los errores de los usuarios. Cuando se define en `true`, la configuración muestra lo que se envía en la red. En el siguiente ejemplo se crea un cliente de Amazon S3 con rastro de red habilitado en el momento de la creación del cliente.

```
s3 = Aws::S3::Client.new(http_wire_trace: true)
```

Dado el código y el argumento `bucket_name` siguientes, la salida muestra un mensaje que indica si existe un bucket con ese nombre.

```
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(client: Aws::S3::Client.new(http_wire_trace: true))

if s3.bucket(ARGV[0]).exists?
  puts "Bucket #{ARGV[0]} exists"
else
  puts "Bucket #{ARGV[0]} does not exist"
end
```

Si el bucket existe, el resultado es similar al siguiente. (Las devoluciones se añaden a la línea HEAD para favorecer la legibilidad).

```
opening connection to bucket_name.s3-us-west-1.amazonaws.com:443...
opened
starting SSL for bucket_name.s3-us-west-1.amazonaws.com:443...
SSL established, protocol: TLSv1.2, cipher: ECDHE-RSA-AES128-GCM-SHA256
-> "HEAD / HTTP/1.1
    Accept-Encoding:
    User-Agent: aws-sdk-ruby3/3.171.0 ruby/3.2.2 x86_64-linux aws-sdk-s3/1.120.0
    Host: bucket_name.s3-us-west-1.amazonaws.com
    X-Amz-Date: 20230427T143146Z
/* omitted */
Accept: */*\r\n\r\n"
-> "HTTP/1.1 200 OK\r\n"
-> "x-amz-id-2: XxB2J+kpHgTjmMUwpkUI1EjaFSPxAjWRgkn/+z7YwWc/
iAX5E3OXRBzJ37cfc8T4D7ELC1KFELM=\r\n"
-> "x-amz-request-id: 5MD4APQQS815QVBR\r\n"
-> "Date: Thu, 27 Apr 2023 14:31:47 GMT\r\n"
-> "x-amz-bucket-region: us-east-1\r\n"
-> "x-amz-access-point-alias: false\r\n"
-> "Content-Type: application/xml\r\n"
-> "Server: AmazonS3\r\n"
-> "\r\n"
Conn keep-alive
Bucket bucket_name exists
```

También puede activar el rastreo de red después de crear el cliente.

```
s3 = Aws::S3::Client.new
s3.config.http_wire_trace = true
```

Para obtener más información sobre los campos de la información del rastreo de red, consulte los encabezados de solicitud obligatorios de Transfer Family.

Disociar respuestas y errores de cliente

Obtenga información sobre cómo disociar respuestas de cliente y errores de cliente en una aplicación AWS SDK para Ruby.

Disociación de respuestas de cliente

Cuando disocia una respuesta, AWS SDK para Ruby deshabilita el tráfico de red y el cliente devuelve datos disociados (o falsos). Si no se suministran datos de disociados, el cliente devuelve:

- Las listas como matrices vacías
- Los mapas como hashes vacíos
- Los valores numéricos como cero
- Las fechas como now

El siguiente ejemplo devuelve nombres disociados para la lista de buckets de Amazon S3.

```
require 'aws-sdk'

s3 = Aws::S3::Client.new(stub_responses: true)

bucket_data = s3.stub_data(:list_buckets, :buckets => [{name:'aws-sdk'}, {name:'aws-
sdk2'}])
s3.stub_responses(:list_buckets, bucket_data)
bucket_names = s3.list_buckets.buckets.map(&:name)

# List each bucket by name
bucket_names.each do |name|
  puts name
end
```

La ejecución de este código muestra lo siguiente.

```
aws-sdk
aws-sdk2
```

> **Note**
>
> Una vez proporcionados datos disociados, se dejan de aplicar los valores predeterminados a los atributos de instancia restantes. Esto significa que, en el ejemplo anterior, el atributo de instancia restante `creation_date` no es now sino `nil`.

AWS SDK para Ruby valida sus datos disociados. Si pasa datos del tipo equivocado, se genera una excepción `ArgumentError`. Por ejemplo, si en lugar de la asignación anterior a `bucket_data`, se hubiera usado lo siguiente:

```
bucket_data = s3.stub_data(:list_buckets, buckets:['aws-sdk', 'aws-sdk2'])
```

AWS SDK para Ruby generara dos excepciones `ArgumentError`.

```
expected params[:buckets][0] to be a hash
expected params[:buckets][1] to be a hash
```

Disociación de errores de cliente

También puede proporcionar datos disociados para los errores que AWS SDK para Ruby genera para métodos específicos. El ejemplo siguiente muestra `Caught Timeout::Error error calling head_bucket on aws-sdk`.

```
require 'aws-sdk'

s3 = Aws::S3::Client.new(stub_responses: true)
s3.stub_responses(:head_bucket, Timeout::Error)

begin
  s3.head_bucket({bucket: 'aws-sdk'})
rescue Exception => ex
  puts "Caught #{ex.class} error calling 'head_bucket' on 'aws-sdk'"
end
```

Paginación

Algunas llamadas a AWS proporcionan respuestas paginadas para limitar la cantidad de datos que se devuelve con cada respuesta. Una página de datos representa un máximo de 1000 elementos.

Las respuestas paginadas se pueden enumerar

La forma más sencilla de administrar datos de respuesta paginados es usar el enumerador integrado en el objeto de respuesta, como se muestra en el siguiente ejemplo.

```
s3 = Aws::S3::Client.new
```

```
s3.list_objects(bucket:'aws-sdk').each do |response|
  puts response.contents.map(&:key)
end
```

Esto produce un objeto de respuesta por cada llamada realizada a la API y enumera los objetos del bucket designado. El SDK recupera páginas adicionales de datos para completar la solicitud.

Administrar respuestas paginadas manualmente

Si desea controlar la paginación por sí mismo, use el método `next_page?` de la respuesta para comprobar si hay más páginas que pueden recuperarse o use el método `last_page?` para verificar que no hay más páginas que puedan recuperarse.

Si hay más páginas, utilice el método `next_page` (observe que no es ?) para recuperar la siguiente página de resultados, como se muestra en el siguiente ejemplo.

```
s3 = Aws::S3::Client.new

# Get the first page of data
response = s3.list_objects(bucket:'aws-sdk')

# Get additional pages
while response.next_page? do
  response = response.next_page
  # Use the response data here...
end
```

> **Note**
>
> Si llama al método `next_page` y no hay más páginas que puedan recuperarse, el SDK genera una excepción Aws::PageableResponse::LastPageError.

Clases de datos paginados

Los datos paginados en AWS SDK para Ruby se administran a través de la clase Aws::PageableResponse, que se incluye con Seahorse::Client::Response para proporcionar acceso a los datos paginados.

Esperadores

Los esperadores son métodos de utilidad que sondean si un determinado estado se produce en un cliente. Los esperadores pueden fallar transcurrido un número de intentos en un intervalo de sondeo definido para el cliente del servicio. Para ver un ejemplo de cómo se usa un esperador, consulte el método create_table de Amazon DynamoDB Encryption Client en el repositorio de ejemplos de código AWS.

Invocación de un esperador

Para invocar un esperador, llame a `wait_until` en el cliente del servicio. En el siguiente ejemplo, un esperador espera hasta que la instancia `i-12345678` se esté ejecutando antes de continuar.

```
ec2 = Aws::EC2::Client.new

begin
  ec2.wait_until(:instance_running, instance_ids:['i-12345678'])
  puts "instance running"
rescue Aws::Waiters::Errors::WaiterFailed => error
  puts "failed waiting for instance running: #{error.message}"
end
```

El primer parámetro corresponde al nombre del esperador, que es específico del cliente del servicio, e indica la operación que se está esperado. El segundo parámetro es un hash de los parámetros que se transfieren al método del cliente que ha llamado el esperador, lo cual varía según el nombre del esperador.

Para obtener una lista de las operaciones a las que el esperador puede esperar y los métodos del cliente que se llaman en cada operación, consulte la documentación de los campos `waiter_names` y `wait_until` para el cliente que esté utilizando.

Errores de espera

Los esperadores pueden fallar con cualquiera de las siguientes excepciones.

Aws::Waiters::Errors::FailureStateError

Se ha producido un estado de error durante la espera.

Aws::Waiters::Errors::NoSuchWaiterError

No se ha definido el nombre del esperador especificado para el cliente que se está utilizando.

Aws::Waiters::Errors::TooManyAttemptsError

El número de intentos ha superado el valor `max_attempts` del esperador.

Aws::Waiters::Errors::UnexpectedError

Se ha producido un error inesperado durante la espera.

Aws::Waiters::Errors::WaiterFailed

Se ha superado uno de los estados de espera o se ha producido otro error durante la espera.

Todos estos errores, excepto `NoSuchWaiterError`, se basan en `WaiterFailed`. Para detectar errores en un esperador, utilice `WaiterFailed`, tal y como se muestra en el siguiente ejemplo.

```
rescue Aws::Waiters::Errors::WaiterFailed => error
  puts "failed waiting for instance running: #{error.message}"
end
```

Configuración de un esperador

Cada esperador tiene un intervalo de sondeo predeterminado y un número máximo de intentos que hará antes de devolver el control al programa. Para establecer estos valores, utilice los parámetros `max_attempts` y `delay:` en la llamada a `wait_until`. En el siguiente ejemplo se espera hasta 25 segundos, realizando un sondeo cada cinco segundos.

```
# Poll for ~25 seconds
client.wait_until(...) do |w|
  w.max_attempts = 5
  w.delay = 5
end
```

Para deshabilitar los errores de espera, establezca el valor de cualquiera de estos parámetros en `nil`.

Ampliación de un esperador

Para modificar el comportamiento de los esperadores, puede registrar las devoluciones que se activan antes de cada intento de sondeo y antes del periodo de espera.

En el siguiente ejemplo se implementa un retardo exponencial en un esperador duplicando el tiempo que se debe esperar en cada intento.

```
ec2 = Aws::EC2::Client.new

ec2.wait_until(:instance_running, instance_ids:['i-12345678']) do |w|
  w.interval = 0 # disable normal sleep
  w.before_wait do |n, resp|
    sleep(n ** 2)
  end
end
```

En el siguiente ejemplo se deshabilita el número máximo de intentos y, en su lugar, se espera una hora (3.600 segundos) antes de que se produzca el error.

```
started_at = Time.now
client.wait_until(...) do |w|
  # Disable max attempts
  w.max_attempts = nil

  # Poll for one hour, instead of a number of attempts
  w.before_wait do |attempts, response|
    throw :failure if Time.now - started_at > 3600
  end
end
```

Especificar el comportamiento de reintento del cliente

De forma predeterminada, AWS SDK para Ruby realiza hasta tres reintentos, dejando transcurrir 15 segundos entre los reintentos, lo que equivale a un total de cuatro intentos. Por lo tanto, una operación podría tardar hasta 60 segundos en agotar el tiempo de espera.

El siguiente ejemplo crea un cliente de Amazon S3 en la región `us-west-2` y especifica esperar cinco segundos entre dos reintentos en cada operación de cliente. Por lo tanto, las operaciones de cliente de Amazon S3 podrían tardar hasta 15 segundos en agotar el tiempo de espera.

```
 s3 = Aws::S3::Client.new(
   region: region,
   retry_limit: 2,
   retry_backoff: lambda { |c| sleep(5) }
)
```

En este ejemplo se muestra cómo cambiar los parámetros de reintento directamente en el código. Sin embargo, también puede usar variables de entorno o el archivo compartido `config` de AWS para configurarlas para su aplicación. Para obtener más información sobre estos ajustes, consulte Comportamiento de reintento en la Guía de referencia de las herramientas y los SDK de AWS. Cualquier configuración explícita establecida en el código o en el propio cliente de servicio tiene prioridad sobre la establecida en las variables de entorno o en el archivo compartido `config`.

Migrar de la versión 1 o 2 a la versión 3 de AWS SDK para Ruby

El objetivo de este tema es ayudar a migrar de la versión 1 o 2 de AWS SDK para Ruby a la versión 3.

Uso en paralelo

No es necesario reemplazar la versión 1 o 2 de AWS SDK para Ruby por la versión 3. Puede utilizarlas de forma conjunta en la misma aplicación. Para obtener más información, consulte esta entrada del blog.

Un ejemplo rápido.

```
require 'aws-sdk-v1' # version 1
require 'aws-sdk'    # version 2
require 'aws-sdk-s3' # version 3

s3 = AWS::S3::Client.new # version 1
s3 = Aws::S3::Client.new # version 2 or 3
```

No es necesario volver a escribir el código funcional de la versión 1 o 2 existente para comenzar a utilizar el SDK de la versión 3. Una estrategia de migración válida es escribir solo el nuevo código con respecto al SDK de la versión 3.

Diferencias generales

La versión 3 difiere de la versión 2 en un aspecto importante.

- Cada servicio está disponible como una gema independiente.

La versión 2 difiere de la versión 1 en varios aspectos importantes.

- El espacio de nombres raíz es distinto: `Aws` frente a `AWS`. Esto permite realizar un uso en paralelo.

- `Aws.config`: ahora es un hash de Ruby simple, en lugar de un método.
- Estrictas opciones del constructor: al construir un cliente o un objeto de recurso en el SDK de la versión 1, no se tienen en cuenta las opciones del constructor desconocidas. En la versión 2, las opciones del constructor desconocidas activan un `ArgumentError`. Por ejemplo:

```
# version 1
AWS::S3::Client.new(http_reed_timeout: 10)
# oops, typo'd option is ignored

# version 2
Aws::S3::Client.new(http_reed_timeout: 10)
# => raises ArgumentError
```

Diferencias del cliente

No existen importantes diferencias entre las clases de cliente de la versión 2 y la versión 3.

Entre la versión 1 y 2, las clases del cliente presentan menos diferencias externas. Muchos clientes de servicios tendrán interfaces compatibles tras la construcción del cliente. Algunas de las diferencias más destacadas:

- `Aws::S3::Client`: en la versión 1, la clase del cliente de Amazon S3 tiene codificación manual. La versión 2 se genera a partir de un modelo de servicio. Los nombres de método y las entradas son muy diferentes en la versión 2.
- `Aws::EC2::Client`: la versión 2 utiliza nombre en plural para las listas de salida, mientras que la versión 1 utiliza el sufijo `_set`. Por ejemplo:

```
# version 1
resp = AWS::EC2::Client.new.describe_security_groups
resp.security_group_set
#=> [...]

# version 2
resp = Aws::EC2::Client.new.describe_security_groups
resp.security_groups
#=> [...]
```

- `Aws::SWF::Client`: la versión 2 utiliza respuestas estructuradas, mientras que la versión 1 utiliza hash de Ruby simples.

- Nombres distintos según la clase de servicio: la versión 2 utiliza un nombre diferente para los distintos servicios:
  - `AWS::SimpleWorkflow` se ha convertido en `Aws::SWF`
  - `AWS::ELB` se ha convertido en `Aws::ElasticLoadBalancing`
  - `AWS::SimpleEmailService` se ha convertido en `Aws::SES`
- Opciones de configuración de clientes: algunas de las opciones de configuración de la versión 1 han cambiado de nombre en la versión 2. Otras se han eliminado o reemplazado. A continuación, se muestran los principales cambios:
  - Se ha eliminado `:use_ssl`. La versión 2 utiliza SSL para todo. Para deshabilitar SSL debe configurar un `:endpoint` que utilice `http://`.
  - `:ssl_ca_file` ahora es `:ssl_ca_bundle`
  - `:ssl_ca_path` ahora es `:ssl_ca_directory`
  - `:ssl_ca_store` añadido.
  - Ahora `:endpoint` debe ser un URI HTTP o HTTPS completo en lugar de un nombre de host.
  - Se han quitado las opciones `:*_port` de cada servicio y ahora se han reemplazado por `:endpoint`.
  - `:user_agent_prefix` ahora es `:user_agent_suffix`

Diferencias en los recursos

No existen importantes diferencias entre las interfaces de recurso de la versión 2 y la versión 3.

Existen importantes diferencias entre las interfaces de recurso de la versión 1 y la versión 2. Toda la versión 1 está codificada manualmente, mientras que las interfaces de recurso de la versión 2 se generan a partir de un modelo. Las interfaces de recurso de la versión 2 son significativamente más coherentes. Algunas de las diferencias entre sistemas son:

- Clase de recurso independiente: en la versión 2, el nombre del servicio es un módulo, no una clase. En este módulo se encuentra la interfaz de recurso:

```
# version 1
s3 = AWS::S3.new

# version 2
s3 = Aws::S3::Resource.new
```

- Referencia a los recursos: el SDK de la versión 2 separa los métodos getter de las colecciones y los recursos individuales en dos métodos diferentes:

```
# version 1
s3.buckets['bucket-name'].objects['key'].delete

# version 2
s3.bucket('bucket-name').object('key').delete
```

- Operaciones por lotes: en la versión 1, todas las operaciones por lotes eran utilidades con codificación manual. En la versión 2, muchas operaciones por lotes son operaciones procesadas por lotes de generación automática a través de la API. Las interfaces procesadas por lotes de la versión 2 son muy distintas de la versión 1.

Trabajar con Servicios de AWS en AWS SDK para Ruby

En las siguientes secciones se ofrecen explicaciones y ejemplos para mostrarle cómo usar AWS SDK para Ruby para trabajar con Servicios de AWS.

Si está empezando con el AWS SDK para Ruby, le recomendamos que lea primero el tema Introducción.

- Ejemplos de código con orientaciones: Proporciona ejemplos guiados para varios Servicios de AWS.
- Ejemplos de código: Proporciona una lista completa de ejemplos de los servicios disponibles (pero sin orientación adicional más allá del código).

El código fuente de todos estos ejemplos está disponible para su descarga en el Repositorio de ejemplos de código de AWS en GitHub. Para proponer un nuevo ejemplo de código para que el equipo de documentación de AWS considere la posibilidad de crearlo, cree una nueva solicitud. El equipo está buscando crear ejemplos de código que abarquen situaciones y casos de uso más amplios, en comparación con fragmentos de código sencillos que tratan solo llamadas a la API individuales. Para obtener instrucciones, consulte la sección Proposing new code examples en el archivo Readme en GitHub.

Ejemplos de código con orientaciones para AWS SDK para Ruby

En esta sección se proporcionan ejemplos que pueden utilizarse para tener acceso a Servicios de AWS mediante AWS SDK para Ruby.

Busque el código fuente de estos ejemplos y otros en el Repositorio de ejemplos de código de AWS en GitHub.

Temas

- Ejemplos de CloudTrail con AWS SDK para Ruby
- Ejemplos del uso de AWS SDK para Ruby en Amazon CloudWatch
- Ejemplos de CodeBuild con AWS SDK para Ruby
- Ejemplos de Amazon EC2 con AWS SDK para Ruby
- Ejemplos de AWS Elastic Beanstalk con AWS SDK para Ruby
- Ejemplos de AWS Identity and Access Management (IAM) con AWS SDK para Ruby

- Ejemplos de AWS Key Management Service con AWS SDK para Ruby
- Ejemplos de AWS Lambda con AWS SDK para Ruby
- Ejemplos de Amazon Polly con AWS SDK para Ruby
- Ejemplos de Amazon RDS con AWS SDK para Ruby
- Ejemplos de Amazon SES con AWS SDK para Ruby
- Ejemplos de Amazon SNS con AWS SDK para Ruby
- Ejemplos de Amazon SQS con AWS SDK para Ruby
- Ejemplos de Amazon WorkDocs

Ejemplos de CloudTrail con AWS SDK para Ruby

CloudTrail es un Servicio de AWS que puede utilizar para monitorizar sus implementaciones de AWS en la nube mediante la obtención de un historial de llamadas a la API de AWS para su cuenta. Puede utilizar los siguientes ejemplos de códigos de AWS SDK para Ruby obtener acceso a AWS CloudTrail. Para obtener más información acerca de CloudTrail, consulte la documentación de AWS CloudTrail;.

Temas

- Obtención de listas de registros de seguimiento de CloudTrail
- Creación de un registro de seguimiento de CloudTrail
- Obtención de listas de eventos de registros de seguimiento de CloudTrail
- Eliminación de un registro de seguimiento de CloudTrail

Obtención de listas de registros de seguimiento de CloudTrail

En este ejemplo se utiliza el método describe_trails para obtener una lista de los nombres de los registros de seguimiento de CloudTrail y el bucket en el que CloudTrail almacena información en la región `us-west-2`.

Elija `Copy` para guardar el código localmente.

Cree el archivo describe_trails.rb con el siguiente código:

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
```

```
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-cloudtrail'  # v2: require 'aws-sdk'

# Create client in us-west-2
client = Aws::CloudTrail::Client.new(region: 'us-west-2')

resp = client.describe_trails({})

puts
puts "Found #{resp.trail_list.count} trail(s) in us-west-2:"
puts

resp.trail_list.each do |trail|
  puts 'Name:           ' + trail.name
  puts 'S3 bucket name: ' + trail.s3_bucket_name
  puts
end
```

Consulte el ejemplo completo en GitHub.

Creación de un registro de seguimiento de CloudTrail

En este ejemplo se utiliza el método create_trail para crear un registro de seguimiento de CloudTrail en la región `us-west-2`. Requiere dos entradas: el nombre del registro de seguimiento y el nombre del bucket en el que CloudTrail almacena información. Si el bucket no tiene la política adecuada, incluya la marca -p para asociar la política correcta al bucket.

Elija `Copy` para guardar el código localmente.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
```

```
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-cloudtrail'  # v2: require 'aws-sdk'
require 'aws-sdk-s3'
require 'aws-sdk-sts'

# Attach IAM policy to bucket
def add_policy(bucket)
  # Get account ID using STS
  sts_client = Aws::STS::Client.new(region: 'us-west-2')
  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to S3 bucket
  s3_client = Aws::S3::Client.new(region: 'us-west-2')

  begin
    policy = {
      'Version' => '2012-10-17',
      'Statement' => [
        {
          'Sid' => 'AWSCloudTrailAclCheck20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com',
          },
          'Action' => 's3:GetBucketAcl',
          'Resource' => 'arn:aws:s3:::' + bucket,
        },
        {
          'Sid' => 'AWSCloudTrailWrite20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com',
          },
          'Action' => 's3:PutObject',
          'Resource' => 'arn:aws:s3:::' + bucket + '/AWSLogs/' + account_id + '/*',
          'Condition' => {
            'StringEquals' => {
              's3:x-amz-acl' => 'bucket-owner-full-control',
```

```
            },
          },
        },
      ]
    }.to_json

    s3_client.put_bucket_policy(
      bucket: bucket,
      policy: policy
    )

    puts 'Successfully added policy to bucket ' + bucket
  rescue StandardError => err
    puts 'Got error trying to add policy to bucket ' + bucket + ':'
    puts err
    exit 1
  end
end

# main
name = ''
bucket = ''
attach_policy = false

i = 0

while i < ARGV.length
  case ARGV[i]
    when '-b'
      i += 1
      bucket = ARGV[i]

    when '-p'
      attach_policy = true

    else
      name = ARGV[i]
  end

  i += 1
end

if name == '' || bucket == ''
  puts 'You must supply a trail name and bucket name'
```

```
  puts USAGE
  exit 1
end

if attach_policy
  add_policy(bucket)
end

# Create client in us-west-2
client = Aws::CloudTrail::Client.new(region: 'us-west-2')

begin
  client.create_trail({
    name: name, # required
    s3_bucket_name: bucket, # required
  })

  puts 'Successfully created CloudTrail ' + name + ' in us-west-2'
rescue StandardError => err
  puts 'Got error trying to create trail ' + name + ':'
  puts err
  exit 1
end
```

Consulte el ejemplo completo en GitHub.

Obtención de listas de eventos de registros de seguimiento de CloudTrail

En este ejemplo se utiliza el método lookup_events para obtener una lista de los eventos de registro de seguimiento de CloudTrail en la región `us-west-2`.

Elija Copy para guardar el código localmente.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
```

```
# language governing permissions and limitations under the License.

require 'aws-sdk-cloudtrail'  # v2: require 'aws-sdk'

def show_event(event)
  puts 'Event name:   ' + event.event_name
  puts 'Event ID:     ' + event.event_id
  puts "Event time:   #{event.event_time}"
  puts 'User name:    ' + event.username

  puts 'Resources:'

  event.resources.each do |r|
    puts '  Name:       ' + r.resource_name
    puts '  Type:       ' + r.resource_type
    puts ''
  end
end

# Create client in us-west-2
client = Aws::CloudTrail::Client.new(region: 'us-west-2')

resp = client.lookup_events()

puts
puts "Found #{resp.events.count} events in us-west-2:"
puts

resp.events.each do |e|
  show_event(e)
end
```

Consulte el ejemplo completo en GitHub.

Eliminación de un registro de seguimiento de CloudTrail

En este ejemplo se utiliza el método delete_trail para eliminar un registro de seguimiento de CloudTrail en la región `us-west-2`. Requiere una entrada, el nombre del registro de seguimiento.

Elija Copy para guardar el código localmente.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
```

```
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-cloudtrail'  # v2: require 'aws-sdk'

if ARGV.length != 1
  puts 'You must supply the name of the trail to delete'
  exit 1
end

name = ARGV[0]

# Create client in us-west-2
client = Aws::CloudTrail::Client.new(region: 'us-west-2')

begin
  client.delete_trail({
    name: name, # required
  })

  puts 'Successfully deleted CloudTrail ' + name + ' in us-west-2'
rescue StandardError => err
  puts 'Got error trying to delete trail ' + name + ':'
  puts err
  exit 1
end
```

Consulte el ejemplo completo en GitHub.

Ejemplos del uso de AWS SDK para Ruby en Amazon CloudWatch

Amazon CloudWatch (CloudWatch) es un servicio de monitorización de recursos de la nube de AWS y las aplicaciones que se ejecutan en AWS. Puede utilizar los siguientes ejemplos para acceder a CloudWatch mediante AWS SDK para Ruby. Para obtener más información sobre CloudWatch, consulte la documentación de Amazon CloudWatch.

Temas

- Obtención de información sobre las alarmas de Amazon CloudWatch
- Creación de una alarma de Amazon CloudWatch
- Activación y desactivación de acciones de alarma de Amazon CloudWatch
- Obtención de información acerca de métricas personalizadas para Amazon CloudWatch
- Envío de eventos a Eventos de Amazon CloudWatch

Obtención de información sobre las alarmas de Amazon CloudWatch

En el siguiente ejemplo de código se muestra información sobre las alarmas métricas disponibles en Amazon CloudWatch.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

require 'aws-sdk-cloudwatch'

# Displays information about available metric alarms in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts '-' * 16
      puts 'Name:           ' + alarm.alarm_name
      puts 'State value:    ' + alarm.state_value
      puts 'State reason:   ' + alarm.state_reason
      puts 'Metric:         ' + alarm.metric_name
      puts 'Namespace:      ' + alarm.namespace
      puts 'Statistic:      ' + alarm.statistic
      puts 'Period:         ' + alarm.period.to_s
      puts 'Unit:           ' + alarm.unit.to_s
      puts 'Eval. periods:  ' + alarm.evaluation_periods.to_s
      puts 'Threshold:      ' + alarm.threshold.to_s
      puts 'Comp. operator: ' + alarm.comparison_operator
```

```
      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts 'OK actions:'
        alarm.ok_actions.each do |a|
          puts '  ' + a
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
        puts 'Alarm actions:'
        alarm.alarm_actions.each do |a|
          puts '  ' + a
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
          alarm.insufficient_data_actions.count.positive?
        puts 'Insufficient data actions:'
        alarm.insufficient_data_actions.each do |a|
          puts '  ' + a
        end
      end

      puts 'Dimensions:'
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts '  Name: ' + d.name + ', Value: ' + d.value
        end
      else
        puts '  None for this alarm.'
      end
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Full example call:
def run_me
  region = ''

  # Print usage information and then stop.
```

```
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby cw-ruby-example-show-alarms.rb REGION'
    puts 'Example: ruby cw-ruby-example-show-alarms.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts 'Available alarms:'
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

Creación de una alarma de Amazon CloudWatch

En el siguiente ejemplo de código se crea una nueva alarma de CloudWatch (o se actualiza una alarma existente, si ya existe una alarma con el nombre especificado).

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

require 'aws-sdk-cloudwatch'

# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
```

```
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
```

```
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  return true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  return false
end

# Full example call:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: 'BucketName',
      value: 'doc-example-bucket'
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
```

```
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Activación y desactivación de acciones de alarma de Amazon CloudWatch

El siguiente ejemplo de código:

1. Crea y activa una nueva alarma de CloudWatch (o actualiza una alarma existente, si ya existe una alarma con el nombre especificado).
2. Desactiva la alarma nueva o existente. Para volver a activar la alarma, llame a `enable_alarm_actions`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# The following code example shows how to:
# 1. Create or update an Amazon CloudWatch alarm.
# 2. Disable all actions for an alarm.

require 'aws-sdk-cloudwatch'

# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
```

```
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  return true
rescue StandardError => e
```

```
  puts "Error creating alarm: #{e.message}"
  return false
end

# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  return true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  return false
end

# Full example call:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: 'BucketName',
      value: 'doc-example-bucket'
    },
    {
```

```
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

  if alarm_actions_disabled?(cloudwatch_client, alarm_name)
    puts "Alarm '#{alarm_name}' disabled."
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Obtención de información acerca de métricas personalizadas para Amazon CloudWatch

El siguiente ejemplo de código:

1. Añade puntos de datos a una métrica personalizada en CloudWatch.
2. Muestra una lista de las métricas disponibles para un espacio de nombres de métricas en CloudWatch.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# The following example shows how to:
# 1. Add a datapoint to a metric in Amazon CloudWatch.
# 2. List available metrics for a metric namespace in Amazon CloudWatch.

require 'aws-sdk-cloudwatch'

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
```

```
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  return true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  return false
end

# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
```

```
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts '    Dimensions:'
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts 'No dimensions found.'
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      'Note that it could take up to 15 minutes for recently-added metrics ' \
      'to become available.'
  end
end

# Full example call:
def run_me
  metric_namespace = 'SITE/TRAFFIC'
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisitors',
    'SiteName',
    'example.com',
    5_885.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
```

```
    'UniqueVisits',
    'SiteName',
    'example.com',
    8_628.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'PageViews',
    'PageURL',
    'example.html',
    18_057.0,
    'Count'
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

Envío de eventos a Eventos de Amazon CloudWatch

En el siguiente ejemplo de código se muestra cómo crear y activar una regla de Eventos de Amazon CloudWatch. Esta regla envía una notificación al tema especificado en Amazon Simple Notification Service (Amazon SNS) cada vez que una instancia disponible en Amazon Elastic Compute Cloud (Amazon EC2) cambia a un estado de ejecución. Además, la información de eventos relacionados se registra en un grupo de registro de CloudWatch Events.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# The following code example shows how to create and trigger a rule in
# Amazon CloudWatch Events. This rule sends a notification to the specified
# topic in Amazon Simple Notification Service (Amazon SNS) whenever an
# available instance in Amazon Elastic Compute Cloud (Amazon EC2) changes
# to a running state. Also, related event information is logged to a log group
# in Amazon CloudWatch Logs.
#
# This code example works with the following AWS resources through the
# following functions:
```

```
#
# - A rule in Amazon CloudWatch Events. See the rule_exists?, rule_found?,
#   create_rule, and display_rule_activity functions.
# - A role in AWS Identity and Access Management (IAM) to allow the rule
#   to work with Amazon CloudWatch Events. See role_exists?, role_found?,
#   and create_role.
# - An Amazon EC2 instance, which triggers the rule whenever it is restarted.
#   See instance_restarted?.
# - A topic and topic subscription in Amazon SNS for the rule to send event
#   notifications to. See topic_exists?, topic_found?, and create_topic.
# - A log group in Amazon CloudWatch Logs to capture related event information.
#   See log_group_exists?, log_group_created?, log_event, and display_log_data.
#
# This code example requires the following AWS resources to exist in advance:
#
# - An Amazon EC2 instance to restart, which triggers the rule.
#
# The run_me function toward the end of this code example calls the
# preceding functions in the correct order.

require 'aws-sdk-sns'
require 'aws-sdk-iam'
require 'aws-sdk-cloudwatchevents'
require 'aws-sdk-ec2'
require 'aws-sdk-cloudwatch'
require 'aws-sdk-cloudwatchlogs'
require 'securerandom'

# Checks whether the specified Amazon Simple Notification Service
# (Amazon SNS) topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The Amazon Resource Name (ARN) of the
#   topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
```

```
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  return false
end

# Checks whether the specified topic exists among those available to the
# caller in Amazon Simple Notification Service (Amazon SNS).
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The Amazon Resource Name (ARN) of the
#   topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts 'Topic found.'
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts 'Topic found.'
          return true
        end
      end
    end
  end
  puts 'Topic not found.'
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
end
```

```
# Creates a topic in Amazon Simple Notification Service (Amazon SNS)
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The Amazon Resource Name (ARN) of the topic that
#   was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: 'email',
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts 'Subscription created with ARN ' \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    'and confirm the subscription to start receiving notification emails.'
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return 'Error'
end

# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The Amazon Resource Name (ARN) of the
#   role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
```

```
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end

# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The Amazon Resource Name (ARN) of the
#   role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts 'Role found.'
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.roles.count.positive?
        if role_found?(response.roles, role_arn)
          puts 'Role found.'
          return true
        end
      end
    end
```

```
  end
  puts 'Role not found.'
  return false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  return false
end

# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon CloudWatch Events to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The Amazon Resource Name (ARN) of the role that
#   was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'
          },
          'Action': 'sts:AssumeRole'
        }
      ]
    }.to_json,
    path: '/',
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts 'Adding access policy to role...'
  iam_client.put_role_policy(
```

```
    policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': 'CloudWatchEventsFullAccess',
          'Effect': 'Allow',
          'Resource': '*',
          'Action': 'events:*'
        },
        {
          'Sid': 'IAMPassRoleForCloudWatchEvents',
          'Effect': 'Allow',
          'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
          'Action': 'iam:PassRole'
        }
      ]
    }.to_json,
    policy_name: 'CloudWatchEventsPolicy',
    role_name: role_name
  )
  puts 'Access policy added to role.'
  return response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts 'If the role was created, you must add the access policy ' \
    'to the role yourself, or delete the role yourself and try again.'
  return 'Error'
end

# Checks whether the specified AWS CloudWatch Events rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
```

```
    return true if rule.name == rule_name
  end
  return false
end

# Checks whether the specified rule exists among those available to the
# caller in AWS CloudWatch Events.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized AWS CloudWatch Events client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts 'Rule found.'
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.rules.count.positive?
        if rule_found?(response.rules, rule_name)
          puts 'Rule found.'
          return true
        end
      end
    end
  end
  puts 'Rule not found.'
  return false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  return false
end

# Creates a rule in AWS CloudWatch Events.
# This rule is triggered whenever an available instance in
```

```
# Amazon Elastic Compute Cloud (Amazon EC2) changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon Simple Notification Service (Amazon SNS).
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized AWS CloudWatch Events client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon Elastic Compute Cloud (Amazon EC2) must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
```

```
      'source': [
        'aws.ec2'
      ],
      'detail-type': [
        'EC2 Instance State-change Notification'
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: 'ENABLED',
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
      put_targets_response.failed_entry_count > 0
    puts 'Error(s) adding target to rule:'
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    return false
  else
    return true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts 'If the rule was created, you must add the target ' \
    'to the rule yourself, or delete the rule yourself and try again.'
  return false
end

# Checks to see whether the specified log group exists among those available
```

```
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts 'Log group found.'
        return true
      end
    end
  end
  puts 'Log group not found.'
  return false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  return false
end

# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
```

```
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts 'Log group created.'
  return true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  return false
end

# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '495426724868310740095796045676567882148068632824696073EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
```

```
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  unless sequence_token.empty?
    event[:sequence_token] = sequence_token
  end

  response = cloudwatchlogs_client.put_log_events(event)
  puts 'Message logged.'
  return response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end

# Restarts an Amazon Elastic Compute Cloud (Amazon EC2) instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
```

```
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ''

  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    'This might take a few minutes...'
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts 'Attempting to restart the instance. This might take a few minutes...'
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance restarted.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  return true
rescue StandardError => e
  puts 'Error creating log stream or stopping or restarting the instance: ' \
    "#{e.message}"
```

```
  log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  return false
end

# Displays information about activity for a rule in Amazon CloudWatch Events.
#
# Prerequisites:
#
# - A rule in Amazon CloudWatch Events.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts 'Attempting to display rule activity...'
  response = cloudwatch_client.get_metric_statistics(
    namespace: 'AWS/Events',
    metric_name: 'Invocations',
    dimensions: [
```

```
      {
        name: 'RuleName',
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ['Sum'],
    unit: 'Count'
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      'specified time period.'
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end

# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts 'Attempting to display log stream data for the log group ' \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
```

```
    log_group_name: log_group_name,
    order_by: 'LastEventTime',
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
      describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts '-' * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
          get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts 'No log messages for this log stream.'
      end
    end
  end
rescue StandardError => e
  puts 'Error getting information about the log streams or their messages: ' \
    "#{e.message}"
end

# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon CloudWatch Events rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
```

```
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts '-' * 10
  puts 'Some of the following AWS resources might still exist in your account.'
  puts 'If you no longer want to use this code example, then to clean up'
  puts 'your AWS account and avoid unexpected costs, you might want to'
  puts 'manually delete any of the following resources if they exist:'
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon CloudWatch Events rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end

# Full example call:
def run_me
  # Properties for the Amazon SNS topic.
  topic_name = 'aws-doc-sdk-examples-topic'
  email_address = 'mary@example.com'
  # Properties for the IAM role.
  role_name = 'aws-doc-sdk-examples-cloudwatch-events-rule-role'
  # Properties for the Amazon CloudWatch Events rule.
  rule_name = 'aws-doc-sdk-examples-ec2-state-change'
  rule_description = 'Triggers when any available EC2 instance starts.'
  instance_state = 'running'
  target_id = 'sns-topic'
  # Properties for the Amazon EC2 instance.
  instance_id = 'i-033c48ef067af3dEX'
  # Properties for displaying the event rule's activity.
  start_time = Time.now - 600 # Go back over the past 10 minutes
                              # (10 minutes * 60 seconds = 600 seconds).
  end_time = Time.now
  period = 60 # Look back every 60 seconds over the past 10 minutes.
  # Properties for the Amazon CloudWatch Logs log group.
  log_group_name = 'aws-doc-sdk-examples-cloudwatch-log'
  # AWS service clients for this code example.
  region = 'us-east-1'
  sts_client = Aws::STS::Client.new(region: region)
  sns_client = Aws::SNS::Client.new(region: region)
  iam_client = Aws::IAM::Client.new(region: region)
  cloudwatchevents_client = Aws::CloudWatchEvents::Client.new(region: region)
  ec2_client = Aws::EC2::Client.new(region: region)
  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
```

```
  cloudwatchlogs_client = Aws::CloudWatchLogs::Client.new(region: region)

  # Get the caller's account ID for use in forming
  # Amazon Resource Names (ARNs) that this code relies on later.
  account_id = sts_client.get_caller_identity.account

  # If the Amazon SNS topic doesn't exist, create it.
  topic_arn = "arn:aws:sns:#{region}:#{account_id}:#{topic_name}"
  unless topic_exists?(sns_client, topic_arn)
    topic_arn = create_topic(sns_client, topic_name, email_address)
    if topic_arn == 'Error'
      puts 'Could not create the Amazon SNS topic correctly. Program stopped.'
      manual_cleanup_notice(
        topic_name, role_name, rule_name, log_group_name, instance_id
      )
      exit 1
    end
  end

  # If the IAM role doesn't exist, create it.
  role_arn = "arn:aws:iam::#{account_id}:role/#{role_name}"
  unless role_exists?(iam_client, role_arn)
    role_arn = create_role(iam_client, role_name)
    if role_arn == 'Error'
      puts 'Could not create the IAM role correctly. Program stopped.'
      manual_cleanup_notice(
        topic_name, role_name, rule_name, log_group_name, instance_id
      )
    end
  end

  # If the Amazon CloudWatch Events rule doesn't exist, create it.
  unless rule_exists?(cloudwatchevents_client, rule_name)
    unless rule_created?(
      cloudwatchevents_client,
      rule_name,
      rule_description,
      instance_state,
      role_arn,
      target_id,
      topic_arn
    )
      puts 'Could not create the Amazon CloudWatch Events rule correctly. ' \
        'Program stopped.'
```

```
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
unless log_group_exists?(cloudwatchlogs_client, log_group_name)
  unless log_group_created?(cloudwatchlogs_client, log_group_name)
    puts 'Could not create the Amazon CloudWatch Logs log group ' \
    'correctly. Program stopped.'
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# Restart the Amazon EC2 instance, which triggers the rule.
unless instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  puts 'Could not restart the instance to trigger the rule. ' \
    'Continuing anyway to show information about the rule and logs...'
end

# Display how many times the rule was triggered over the past 10 minutes.
display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)

# Display related log data in Amazon CloudWatch Logs.
display_log_data(cloudwatchlogs_client, log_group_name)

# Reminder the caller to clean up any AWS resources that are used
# by this code example and are no longer needed.
manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
```

```
  )
end

run_me if $PROGRAM_NAME == __FILE__
```

Ejemplos de CodeBuild con AWS SDK para Ruby

CodeBujild es un servicio de creación completamente administrado que compila código fuente, ejecuta pruebas y produce paquetes de software listos para su implementación. Puede utilizar los siguientes ejemplos de códigos de AWS SDK para Ruby obtener acceso a AWS CodeBuild. Para obtener más información acerca de CodeBuild, consulte la documentación de AWS CodeBuild.

Temas

- Obtención de información acerca de todos los proyectos de AWS CodeBuild
- Compilación de un proyecto de AWS CodeBuild
- Obtención de listas de compilaciones de proyectos de AWS CodeBuild

Obtención de información acerca de todos los proyectos de AWS CodeBuild

En el siguiente ejemplo se muestran los nombres de hasta 100 de los proyectos de AWS CodeBuild.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-codebuild'  # v2: require 'aws-sdk'

client = Aws::CodeBuild::Client.new(region: 'us-west-2')

resp = client.list_projects({
  sort_by: 'NAME', # accepts NAME, CREATED_TIME, LAST_MODIFIED_TIME
  sort_order: 'ASCENDING' # accepts ASCENDING, DESCENDING
```

```
})

resp.projects.each { |p| puts p }

puts
```

Elija Copy para guardar el código localmente. Consulte el ejemplo completo en GitHub.

Compilación de un proyecto de AWS CodeBuild

En el siguiente ejemplo se crea el proyecto de AWS CodeBuild especificado en la línea de comandos. Si no se proporciona ningún argumento de línea de comando, emite un error y sale.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-codebuild'  # v2: require 'aws-sdk'

project_name = ''

if ARGV.length != 1
  puts 'You must supply the name of the project to build'
  exit 1
else
  project_name = ARGV[0]
end

client = Aws::CodeBuild::Client.new(region: 'us-west-2')

begin
  client.start_build(project_name: project_name)
  puts 'Building project ' + project_name
rescue StandardError => ex
  puts 'Error building project: ' + ex.message
```

```
end
```

Elija Copy para guardar el código localmente. Consulte el ejemplo completo en GitHub.

Obtención de listas de compilaciones de proyectos de AWS CodeBuild

En el siguiente ejemplo se muestra información acerca de las compilaciones de proyectos de AWS CodeBuild. Esta información incluye el nombre del proyecto, cuándo comenzó la compilación y cuánto tiempo tardó cada fase de la compilación, en segundos.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-codebuild'  # v2: require 'aws-sdk'

client = Aws::CodeBuild::Client.new(region: 'us-west-2')

build_list = client.list_builds({sort_order: 'ASCENDING', })

builds = client.batch_get_builds({ids: build_list.ids})

builds.builds.each do |build|
  puts 'Project:    ' + build.project_name
  puts 'Phase:      ' + build.current_phase
  puts 'Status:     ' + build.build_status
end
```

Elija Copy para guardar el código localmente. Consulte el ejemplo completo en GitHub.

Ejemplos de Amazon EC2 con AWS SDK para Ruby

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad de computación redimensionable, literalmente, en servidores de los centros de datos de Amazon, que

puede usar para crear y alojar sus sistemas de software. Puede utilizar los siguientes ejemplos para acceder a Amazon EC2 mediante AWS SDK para Ruby. Para obtener más información acerca de Amazon EC2, consulte la documentación de Amazon EC2.

Temas

Creación de una Amazon EC2 VPC

En el siguiente ejemplo de código, se crea una nube privada virtual (VPC) en Amazon Virtual Private Cloud (Amazon VPC) y, seguidamente, se etiqueta la VPC.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'
```

```
# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     '10.0.0.0/24',
#     'my-key',
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "#{e.message}"
  return false
end

# Full example call:
def run_me
  cidr_block = ''
```

```
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-vpc.rb ' \
      'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = '10.0.0.0/24'
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,
    cidr_block,
    tag_key,
    tag_value
  )
    puts 'VPC created and tagged.'
  else
    puts 'VPC not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Creación de una puerta de enlace de Internet y su asociación a una VPC en Amazon EC2

En el siguiente ejemplo de código, se crea una puerta de enlace de Internet que seguidamente se asocia a una nube privada virtual (VPC) en Amazon Virtual Private Cloud (Amazon VPC).

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Creates an internet gateway and then attaches it to a virtual private cloud
# (VPC) in Amazon Virtual Private Cloud (Amazon VPC).
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC to attach the internet gateway.
# @param tag_key [String] The key of the tag to attach to the internet gateway.
# @param tag_value [String] The value of the tag to attach to the
#   internet gateway.
# @return [Boolean] true if the internet gateway was created and attached;
#   otherwise, false.
# @example
#   exit 1 unless internet_gateway_created_and_attached?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'vpc-6713dfEX'
#   )
def internet_gateway_created_and_attached?(
  ec2_resource,
  vpc_id,
  tag_key,
  tag_value
)
  igw = ec2_resource.create_internet_gateway
  puts "The internet gateway's ID is '#{igw.id}'."
  igw.attach_to_vpc(vpc_id: vpc_id)
  igw.create_tags(
    tags: [
      {
```

```
        key: tag_key,
        value: tag_value
      }
    ]
  )
  return true
rescue StandardError => e
  puts "Error creating or attaching internet gateway: #{e.message}"
  puts 'If the internet gateway was created but not attached, you should ' \
    'clean up by deleting the internet gateway.'
  return false
end

# Full example call:
def run_me
  vpc_id = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-attach-igw-vpc.rb ' \
      'VPC_ID TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-attach-igw-vpc.rb ' \
      'vpc-6713dfEX my-key my-value us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if internet_gateway_created_and_attached?(
    ec2_resource,
```

```
    vpc_id,
    tag_key,
    tag_value
  )
    puts "Created and attached internet gateway to VPC '#{vpc_id}'."
  else
    puts "Could not create or attach internet gateway to VPC '#{vpc_id}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Creación de una subred pública para amazon EC2

En el siguiente ejemplo de código, se crea una subred dentro de una nube privada virtual (VPC) en Amazon Virtual Private Cloud (Amazon VPC) y, seguidamente, se etiqueta la subred.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
```

```
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-east-1a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  return false
end

# Full example call:
def run_me
  vpc_id = ''
  cidr_block = ''
  availability_zone = ''
  tag_key = ''
  tag_value = ''
```

```
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-subnet.rb ' \
      'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
      'vpc-6713dfEX 10.0.0.0/24 us-east-1a my-key my-value us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    cidr_block = '10.0.0.0/24'
    availability_zone = 'us-east-1a'
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
    tag_key = ARGV[3]
    tag_value = ARGV[4]
    region = ARGV[5]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if subnet_created_and_tagged?(
    ec2_resource,
    vpc_id,
    cidr_block,
    availability_zone,
    tag_key,
    tag_value
  )
    puts 'Subnet created and tagged.'
  else
    puts 'Subnet not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Creación de una tabla de enrutamiento de Amazon EC2 y asociación a una subred

En el siguiente ejemplo de código se crea una tabla de enrutamiento en Amazon Virtual Private Cloud (Amazon VPC) que, a continuación, se asocia a una subred de Amazon VPC.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Creates a route table in Amazon Virtual Private Cloud (Amazon VPC)
# and then associates the route table with a subnet in Amazon VPC.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
```

```
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts 'Added tags to route table.'
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts 'Created route with destination CIDR block ' \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  return true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  return false
end

# Full example call:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
```

```
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
      'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
      'TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
      'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
      '\'0.0.0.0/0\' my-key my-value us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-0b6f769731EXAMPLE'
    subnet_id = 'subnet-03d9303b57EXAMPLE'
    gateway_id = 'igw-06ca90c011EXAMPLE'
    destination_cidr_block = '0.0.0.0/0'
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    subnet_id = ARGV[1]
    gateway_id = ARGV[2]
    destination_cidr_block = ARGV[3]
    tag_key = ARGV[4]
    tag_value = ARGV[5]
    region = ARGV[6]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if route_table_created_and_associated?(
    ec2_resource,
    vpc_id,
    subnet_id,
    gateway_id,
    destination_cidr_block,
    tag_key,
    tag_value
  )
    puts 'Route table created and associated.'
  else
    puts 'Route table not created or not associated.'
  end
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

Uso de direcciones IP elásticas en Amazon EC2

El siguiente ejemplo de código:

1. Muestra información sobre las direcciones asociadas a una instancia de Amazon Elastic Compute Cloud (Amazon EC2).
2. Crea una dirección IP elástica en Amazon Virtual Private Cloud (Amazon VPC).
3. Asocia la dirección a la instancia.
4. Vuelve a mostrar información sobre las direcciones asociadas a la instancia Esta vez, debería mostrarse la nueva asociación de direcciones.
5. Publica la dirección.
6. Vuelve a mostrar información sobre las direcciones asociadas a la instancia Esta vez, no debería mostrarse la dirección publicada.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

# This code example does the following:
# 1. Displays information about any addresses associated with an
#    Amazon Elastic Compute Cloud (Amazon EC2) instance.
# 2. Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
# 3. Associates the address with the instance.
# 4. Displays information again about addresses associated with the instance.
#    This time, the new address association should display.
# 5. Releases the address.
# 6. Displays information again about addresses associated with the instance.
#    This time, the released address should not display.

require 'aws-sdk-ec2'

# Checks whether the specified Amazon Elastic Compute Cloud
# (Amazon EC2) instance exists.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
```

```
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance exists; otherwise, false.
# @example
#   exit 1 unless instance_exists?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_exists?(ec2_client, instance_id)
  ec2_client.describe_instances(instance_ids: [instance_id])
  return true
rescue StandardError
  return false
end

# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-east-1'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  return response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  return 'Error'
end

# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
```

```
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id,
  )
  return response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  return 'Error'
end

# Gets information about addresses associated with an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @example
#   describe_addresses_for_instance(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def describe_addresses_for_instance(ec2_client, instance_id)
  response = ec2_client.describe_addresses(
    filters: [
      {
        name: 'instance-id',
        values: [instance_id]
      }
    ]
  )
  addresses = response.addresses
```

```
  if addresses.count.zero?
    puts 'No addresses.'
  else
    addresses.each do |address|
      puts '-' * 20
      puts "Public IP:  #{address.public_ip}"
      puts "Private IP: #{address.private_ip_address}"
    end
  end
rescue StandardError => e
  puts "Error getting address information for instance: #{e.message}"
end

# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  return true
rescue StandardError => e
  return "Error releasing Elastic IP address: #{e.message}"
  return false
end

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
```

```
  puts 'Usage:   ruby ec2-ruby-example-elastic-ips.rb ' \
    'INSTANCE_ID REGION'
  puts 'Example: ruby ec2-ruby-example-elastic-ips.rb ' \
    'i-033c48ef067af3dEX us-east-1'
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  instance_id = 'i-033c48ef067af3dEX'
  region = 'us-east-1'
# Otherwise, use the values as specified at the command prompt.
else
  instance_id = ARGV[0]
  region = ARGV[1]
end

ec2_client = Aws::EC2::Client.new(region: region)

unless instance_exists?(ec2_client, instance_id)
  puts "Cannot find instance with ID '#{instance_id}'. Stopping program."
  exit 1
end

puts "Addresses for instance with ID '#{instance_id}' before allocating " \
  'Elastic IP address:'
describe_addresses_for_instance(ec2_client, instance_id)

puts 'Allocating Elastic IP address...'
allocation_id = allocate_elastic_ip_address(ec2_client)
if allocation_id.start_with?('Error')
  puts 'Stopping program.'
  exit 1
else
  puts "Elastic IP address created with allocation ID '#{allocation_id}'."
end

puts 'Associating Elastic IP address with instance...'
association_id = associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
if association_id.start_with?('Error')
  puts 'Stopping program. You must associate the Elastic IP address yourself.'
  exit 1
```

```
  else
    puts 'Elastic IP address associated with instance with association ID ' \
      "'#{association_id}'."
  end

  puts 'Addresses for instance after allocating Elastic IP address:'
  describe_addresses_for_instance(ec2_client, instance_id)

  puts 'Releasing the Elastic IP address from the instance...'
  if elastic_ip_address_released?(ec2_client, allocation_id) == false
    puts 'Stopping program. You must release the Elastic IP address yourself.'
    exit 1
  else
    puts 'Address released.'
  end

  puts 'Addresses for instance after releasing Elastic IP address:'
  describe_addresses_for_instance(ec2_client, instance_id)
end

run_me if $PROGRAM_NAME == __FILE__
```

Creación de un grupo de seguridad de Amazon EC2

En el siguiente ejemplo de código se crea un grupo de seguridad de Amazon EC2 y, a continuación, se añade una regla de salida a ese grupo.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group and
# then adds an outbound rule to that security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon EC2 resource object.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
```

```
# @param vpc_id [String] The ID of the VPC for the security group.
# @param protocol [String] The network protocol for the outbound rule.
# @param from_port [String] The originating port for the outbound rule.
# @param to_port [String] The destination port for the outbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the outbound rule.
# @return [Boolean] true if the security group was created and the outbound
#   rule was added; otherwise, false.
# @example
#   exit 1 unless security_group_created_with_egress?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX',
#     'tcp',
#     '22',
#     '22',
#     '0.0.0.0/0'
#   )
def security_group_created_with_egress?(
  ec2_resource,
  group_name,
  description,
  vpc_id,
  ip_protocol,
  from_port,
  to_port,
  cidr_ip_range
)
  security_group = ec2_resource.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.id}' in VPC with ID '#{vpc_id}'."
  security_group.authorize_egress(
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
```

```
          }
        ]
      }
    ]
  )
  puts "Granted egress to security group '#{group_name}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  return true
rescue StandardError => e
  puts "Error creating security group or granting egress: #{e.message}"
  return false
end

# Full example call:
def run_me
  group_name = ''
  description = ''
  vpc_id = ''
  ip_protocol = ''
  from_port = ''
  to_port = ''
  cidr_ip_range = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-security-group.rb ' \
      'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL FROM_PORT TO_PORT ' \
      'CIDR_IP_RANGE REGION'
    puts 'Example: ruby ec2-ruby-example-create-security-group.rb ' \
      'my-security-group \'This is my security group.\' vpc-6713dfEX ' \
      'tcp 22 22 \'0.0.0.0/0\' us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    group_name = 'my-security-group'
    description = 'This is my security group.'
    vpc_id = 'vpc-6713dfEX'
    ip_protocol = 'tcp'
    from_port = '22'
    to_port = '22'
    cidr_ip_range = '0.0.0.0/0'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
```

```
  else
    group_name = ARGV[0]
    description = ARGV[1]
    vpc_id = ARGV[2]
    ip_protocol = ARGV[3]
    from_port = ARGV[4]
    to_port = ARGV[5]
    cidr_ip_range = ARGV[6]
    region = ARGV[7]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if security_group_created_with_egress?(
    ec2_resource,
    group_name,
    description,
    vpc_id,
    ip_protocol,
    from_port,
    to_port,
    cidr_ip_range
  )
    puts 'Security group created and egress granted.'
  else
    puts 'Security group not created or egress not granted.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Trabajo con grupos de seguridad en Amazon EC2

El siguiente ejemplo:

1. Crea un grupo de seguridad de Amazon EC2
2. Agrega reglas de entrada al grupo de seguridad
3. Muestra información sobre los grupos de seguridad disponibles.
4. Elimina el grupo de seguridad.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX - License - Identifier: Apache - 2.0

# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  return security_group.group_id
rescue StandardError => e
```

```
  puts "Error creating security group: #{e.message}"
  return 'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client,
  security_group_id,
  ip_protocol,
  from_port,
  to_port,
  cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
```

```
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  return true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  return false
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - A security group with inbound rules, outbound rules, or both.
#
# @param p [Aws::EC2::Types::IpPermission] The IP permissions set.
# @example
#   ec2_client = Aws::EC2::Client.new(region: 'us-east-1')
#   response = ec2_client.describe_security_groups
#   unless sg.ip_permissions.empty?
#     describe_security_group_permissions(
#       response.security_groups[0].ip_permissions[0]
#     )
#   end
def describe_security_group_permissions(perm)
  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"

  unless perm.from_port.nil?
    if perm.from_port == '-1' || perm.from_port == -1
      print ', From: All'
    else
      print ", From: #{perm.from_port}"
    end
  end

  unless perm.to_port.nil?
    if perm.to_port == '-1' || perm.to_port == -1
```

```
      print ', To: All'
    else
      print ", To: #{perm.to_port}"
    end
  end

  if perm.key?(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?
    print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}"
  end

  if perm.key?(:ip_ranges) && perm.ip_ranges.count.positive?
    print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}"
  end

  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @example
#   describe_security_groups(Aws::EC2::Client.new(region: 'us-east-1'))
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      puts '-' * (sg.group_name.length + 13)
      puts "Name:        #{sg.group_name}"
      puts "Description: #{sg.description}"
      puts "Group ID:    #{sg.group_id}"
      puts "Owner ID:    #{sg.owner_id}"
      puts "VPC ID:      #{sg.vpc_id}"

      if sg.tags.count.positive?
        puts 'Tags:'
        sg.tags.each do |tag|
          puts "  Key: #{tag.key}, Value: #{tag.value}"
        end
      end

      unless sg.ip_permissions.empty?
        puts 'Inbound rules:' if sg.ip_permissions.count.positive?
```

```
        sg.ip_permissions.each do |p|
          describe_security_group_permissions(p)
        end
      end

      unless sg.ip_permissions_egress.empty?
        puts 'Outbound rules:' if sg.ip_permissions.count.positive?
        sg.ip_permissions_egress.each do |p|
          describe_security_group_permissions(p)
        end
      end
    end
  else
    puts 'No security groups found.'
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param security_group_id [String] The ID of the security group to delete.
# @return [Boolean] true if the security group was deleted; otherwise, false.
# @example
#   exit 1 unless security_group_deleted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'sg-030a858e078f1b9EX'
#   )
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  return true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  return false
end
```

```
# Full example call:
def run_me
  group_name = ''
  description = ''
  vpc_id = ''
  ip_protocol_http = ''
  from_port_http = ''
  to_port_http = ''
  cidr_ip_range_http = ''
  ip_protocol_ssh = ''
  from_port_ssh = ''
  to_port_ssh = ''
  cidr_ip_range_ssh = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-security-group.rb ' \
      'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
      'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
      'CIDR_IP_RANGE_2 REGION'
    puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
      'my-security-group \'This is my security group.\' vpc-6713dfEX ' \
      'tcp 80 80 \'0.0.0.0/0\' tcp 22 22 \'0.0.0.0/0\' us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    group_name = 'my-security-group'
    description = 'This is my security group.'
    vpc_id = 'vpc-6713dfEX'
    ip_protocol_http = 'tcp'
    from_port_http = '80'
    to_port_http = '80'
    cidr_ip_range_http = '0.0.0.0/0'
    ip_protocol_ssh = 'tcp'
    from_port_ssh = '22'
    to_port_ssh = '22'
    cidr_ip_range_ssh = '0.0.0.0/0'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    group_name = ARGV[0]
    description = ARGV[1]
    vpc_id = ARGV[2]
    ip_protocol_http = ARGV[3]
```

```
  from_port_http = ARGV[4]
  to_port_http = ARGV[5]
  cidr_ip_range_http = ARGV[6]
  ip_protocol_ssh = ARGV[7]
  from_port_ssh = ARGV[8]
  to_port_ssh = ARGV[9]
  cidr_ip_range_ssh = ARGV[10]
  region = ARGV[11]
end

security_group_id = ''
security_group_exists = false
ec2_client = Aws::EC2::Client.new(region: region)

puts 'Attempting to create security group...'
security_group_id = create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
if security_group_id == 'Error'
  puts 'Could not create security group. Skipping this step.'
else
  security_group_exists = true
end

if security_group_exists
  puts 'Attempting to add inbound rules to security group...'
  unless security_group_ingress_authorized?(
    ec2_client,
    security_group_id,
    ip_protocol_http,
    from_port_http,
    to_port_http,
    cidr_ip_range_http
  )
    puts 'Could not add inbound HTTP rule to security group. ' \
      'Skipping this step.'
  end

  unless security_group_ingress_authorized?(
    ec2_client,
    security_group_id,
```

```
      ip_protocol_ssh,
      from_port_ssh,
      to_port_ssh,
      cidr_ip_range_ssh
    )
      puts 'Could not add inbound SSH rule to security group. ' \
        'Skipping this step.'
    end
  end

  puts "\nInformation about available security groups:"
  describe_security_groups(ec2_client)

  if security_group_exists
    puts "\nAttempting to delete security group..."
    unless security_group_deleted?(ec2_client, security_group_id)
      puts 'Could not delete security group. You must delete it yourself.'
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Trabajo con pares de claves de Amazon EC2

El siguiente ejemplo de código:

1. Crea un par de claves en Amazon EC2.
2. Muestra información sobre pares de claves disponibles.
3. Elimina el par de claves.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require 'aws-sdk-ec2'
```

```
# Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2) and
# saves the resulting RSA private key file locally in the calling
# user's home directory.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, key_pair_name + '.pem')
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  return true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  return false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  return false
end

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-east-1'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts 'No key pairs found.'
  else
    puts 'Key pair names:'
    result.key_pairs.each do |key_pair|
```

```
      puts key_pair.key_name
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  return true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  return false
end

# Full example call:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
```

```
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'Displaying existing key pair names before creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Creating key pair...'
  unless key_pair_created?(ec2_client, key_pair_name)
    puts 'Stopping program.'
    exit 1
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Deleting key pair...'
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts 'Stopping program. You must delete the key pair yourself.'
    exit 1
  end
  puts 'Key pair deleted.'

  puts '-' * 10
  puts 'Now that the key pair is deleted, ' \
    'also deleting the related private key pair file...'
  filename = File.join(Dir.home, key_pair_name + '.pem')
  File.delete(filename)
  if File.exist?(filename)
    puts "Could not delete file at '#{filename}'. You must delete it yourself."
  else
    puts 'File deleted.'
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after deleting this key pair...'
  describe_key_pairs(ec2_client)
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

Obtención de información acerca de todas las instancias de Amazon EC2

En el siguiente ejemplo de código se muestran los ID y los estados actuales de las instancias de Amazon EC2.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Lists the IDs and current states of available
# Amazon Elastic Compute Cloud (Amazon EC2) instances.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-east-1'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

#Full example call:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
```

```
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

Obtención de información acerca de todas las instancias de Amazon EC2 con un valor de etiqueta específico

En el siguiente ejemplo de código se muestran los ID y los estados actuales de las instancias de Amazon EC2 que se corresponden con la clave y el valor de etiqueta especificados.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Lists the IDs, current states, and tag keys/values of matching
# available Amazon Elastic Compute Cloud (Amazon EC2) instances.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @param tag_key [String] The key portion of the tag to search on.
# @param tag_value [String] The value portion of the tag to search on.
# @example
#   list_instance_ids_states_by_tag(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'my-key',
#     'my-value'
#   )
def list_instance_ids_states_by_tag(ec2_resource, tag_key, tag_value)
  response = ec2_resource.instances(
    filters: [
      {
        name: "tag:#{tag_key}",
        values: [tag_value]
      }
    ]
  )
  if response.count.zero?
```

```
    puts 'No matching instances found.'
  else
    puts 'Matching instances -- ID, state, tag key/value:'
    response.each do |instance|
      print "#{instance.id}, #{instance.state.name}"
      instance.tags.each do |tag|
        print ", #{tag.key}/#{tag.value}"
      end
      print "\n"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

#Full example call:
def run_me
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-get-instance-info-by-tag.rb ' \
      'TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-get-instance-info-by-tag.rb ' \
      'my-key my-value us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    tag_key = ARGV[0]
    tag_value = ARGV[1]
    region = ARGV[2]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states_by_tag(ec2_resource, tag_key, tag_value)
end

run_me if $PROGRAM_NAME == __FILE__
```

Obtención de información acerca de una instancia de Amazon EC2 específica

En el siguiente ejemplo se muestra el estado de la instancia de Amazon EC2 especificada.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Lists the state of an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @example
#   list_instance_state(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def list_instance_state(ec2_client, instance_id)
  response = ec2_client.describe_instances(
    instance_ids: [instance_id]
  )
  if response.count.zero?
    puts 'No matching instance found.'
  else
    instance = response.reservations[0].instances[0]
    puts "The instance with ID '#{instance_id}' is '#{instance.state.name}'."
  end
rescue StandardError => e
  puts "Error getting information about instance: #{e.message}"
end

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-list-state-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION'
```

```
    puts 'Example: ruby ec2-ruby-example-list-state-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)
  list_instance_state(ec2_client, instance_id)
end

run_me if $PROGRAM_NAME == __FILE__
```

Creación de una instancia de Amazon EC2

En el siguiente ejemplo de código se crea y etiqueta una instancia de Amazon EC2.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'
require 'base64'

# Creates and tags an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An EC2 key pair.
# - If you want to run any commands on the instance after it starts, a
#   file containing those commands.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @param image_id [String] The ID of the target Amazon Machine Image (AMI).
# @param key_pair_name [String] The name of the existing EC2 key pair.
# @param tag_key [String] The key portion of the tag for the instance.
# @param tag_value [String] The value portion of the tag for the instance.
# @param instance_type [String] The ID of the type of instance to create.
```

```
#   If not specified, the default value is 't2.micro'.
# @param user_data_file [String] The path to the file containing any commands
#   to run on the instance after it starts. If not specified, the default
#   value is an empty string.
# @return [Boolean] true if the instance was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless instance_created?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'ami-0947d2ba12EXAMPLE',
#     'my-key-pair',
#     'my-key',
#     'my-value',
#     't2.micro',
#     'my-user-data.txt'
#   )
def instance_created?(
  ec2_resource,
  image_id,
  key_pair_name,
  tag_key,
  tag_value,
  instance_type = 't2.micro',
  user_data_file = ''
)
  encoded_script = ''

  unless user_data_file == ''
    script = File.read(user_data_file)
    encoded_script = Base64.encode64(script)
  end

  instance = ec2_resource.create_instances(
    image_id: image_id,
    min_count: 1,
    max_count: 1,
    key_name: key_pair_name,
    instance_type: instance_type,
    user_data: encoded_script
  )

  puts 'Creating instance...'

  # Check whether the new instance is in the "running" state.
```

```
  polls = 0
  loop do
    polls += 1
    response = ec2_resource.client.describe_instances(
      instance_ids: [
        instance.first.id
      ]
    )
    # Stop polling after 10 minutes (40 polls * 15 seconds per poll) if not running.
    break if response.reservations[0].instances[0].state.name == 'running' || polls >
 40

    sleep(15)
  end

  puts "Instance created with ID '#{instance.first.id}'."

  instance.batch_create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts 'Instance tagged.'

  return true
rescue StandardError => e
  puts "Error creating or tagging instance: #{e.message}"
  return false
end

# Full example call:
def run_me
  image_id = ''
  key_pair_name = ''
  tag_key = ''
  tag_value = ''
  instance_type = ''
  region = ''
  user_data_file = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
```

```
    puts 'Usage: ruby ec2-ruby-example-create-instance.rb ' \
      'IMAGE_ID KEY_PAIR_NAME TAG_KEY TAG_VALUE INSTANCE_TYPE ' \
      'REGION [USER_DATA_FILE]'
    puts 'Example: ruby ec2-ruby-example-create-instance.rb ' \
      'ami-0947d2ba12EXAMPLE my-key-pair my-key my-value t2.micro ' \
      'us-east-1 my-user-data.txt'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    image_id = 'ami-0947d2ba12EXAMPLE'
    key_pair_name = 'my-key-pair'
    tag_key = 'my-key'
    tag_value = 'my-value'
    instance_type = 't2.micro'
    region = 'us-east-1'
    user_data_file = 'my-user-data.txt'
  # Otherwise, use the values as specified at the command prompt.
  else
    image_id = ARGV[0]
    key_pair_name = ARGV[1]
    tag_key = ARGV[2]
    tag_value = ARGV[3]
    instance_type = ARGV[4]
    region = ARGV[5]
    user_data_file = ARGV[6] if ARGV.count == 7 # If user data file specified.
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if instance_created?(
    ec2_resource,
    image_id,
    key_pair_name,
    tag_key,
    tag_value,
    instance_type,
    user_data_file
  )
    puts 'Created and tagged instance.'
  else
    puts 'Could not create or tag instance.'
  end
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

Detención de una instancia de Amazon EC2

En el siguiente ejemplo se intenta detener una instancia de Amazon EC2 especificada.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Attempts to stop an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'stopping'
      puts 'The instance is already stopping.'
      return true
    when 'stopped'
      puts 'The instance is already stopped.'
      return true
    when 'terminated'
      puts 'Error stopping instance: ' \
        'the instance is terminated, so you cannot stop it.'
      return false
    end
  end
```

```
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return false
end

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  unless instance_stopped?(ec2_client, instance_id)
    puts 'Could not stop instance.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Inicio de una instancia de Amazon EC2

En el siguiente ejemplo se trata de iniciar una instancia de Amazon EC2 especificada.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Attempts to start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was started; otherwise, false.
# @example
#   exit 1 unless instance_started?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'pending'
      puts 'Error starting instance: the instance is pending. Try again later.'
      return false
    when 'running'
      puts 'The instance is already running.'
      return true
    when 'terminated'
      puts 'Error starting instance: ' \
        'the instance is terminated, so you cannot start it.'
      return false
    end
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
```

```
  puts 'Instance started.'
  return true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  return false
end

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to start instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  unless instance_started?(ec2_client, instance_id)
    puts 'Could not start instance.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Reinicio de una instancia de Amazon EC2

En el siguiente ejemplo se intenta reiniciar una instancia de Amazon EC2 especificada.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Reboots an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @example
#   request_instance_reboot(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def request_instance_reboot(ec2_client, instance_id)
  response = ec2_client.describe_instances(instance_ids: [instance_id])
  if response.count.zero?
    puts 'Error requesting reboot: no matching instance found.'
  else
    instance = response.reservations[0].instances[0]
    if instance.state.name == 'terminated'
      puts 'Error requesting reboot: the instance is already terminated.'
    else
      ec2_client.reboot_instances(instance_ids: [instance_id])
      puts 'Reboot request sent.'
    end
  end
rescue StandardError => e
  puts "Error requesting reboot: #{e.message}"
end

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-reboot-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION'
```

```
    puts 'Example: ruby ec2-ruby-example-reboot-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)
  request_instance_reboot(ec2_client, instance_id)
end

run_me if $PROGRAM_NAME == __FILE__
```

Administración de instancias de Amazon EC2

El siguiente ejemplo de código:

1. Detiene una instancia de Amazon EC2.
2. Vuelve a iniciar la instancia.
3. Reinicia la instancia.
4. Habilita la monitorización detallada para la instancia.
5. Muestra información sobre instancias disponibles.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

# This code example does the following:
# 1. Stops an Amazon Elastic Compute Cloud (Amazon EC2) instance.
# 2. Restarts the instance.
# 3. Reboots the instance.
# 4. Enables detailed monitoring for the instance.
# 5. Displays information about available instances.

require 'aws-sdk-ec2'
```

```
# Waits for an Amazon Elastic Compute Cloud (Amazon EC2) instance
# to reach the specified state.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_state [Symbol] The desired instance state.
# @param instance_id [String] The ID of the instance.
# @example
#   wait_for_instance(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     :instance_stopped,
#     'i-033c48ef067af3dEX'
#   )
def wait_for_instance(ec2_client, instance_state, instance_id)
  ec2_client.wait_until(instance_state, instance_ids: [instance_id])
  puts "Success: #{instance_state}."
rescue Aws::Waiters::Errors::WaiterFailed => e
  puts "Failed: #{e.message}"
end

# Attempts to stop an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_stopped?(ec2_client, instance_id)
  ec2_client.stop_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_stopped, instance_id)
  return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return false
```

```
end

# Attempts to restart an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was restarted; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_restarted?(ec2_client, instance_id)
  ec2_client.start_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_running, instance_id)
  return true
rescue StandardError => e
  puts "Error restarting instance: #{e.message}"
  return false
end

# Attempts to reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was rebooted; otherwise, false.
# @example
#   exit 1 unless instance_rebooted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_rebooted?(ec2_client, instance_id)
  ec2_client.reboot_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_status_ok, instance_id)
  return true
rescue StandardError => e
```

```
  puts "Error rebooting instance: #{e.message}"
  return false
end

# Attempts to enabled detailed monitoring for an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if detailed monitoring was enabled; otherwise, false.
# @example
#   exit 1 unless instance_detailed_monitoring_enabled?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_detailed_monitoring_enabled?(ec2_client, instance_id)
  result = ec2_client.monitor_instances(instance_ids: [instance_id])
  puts "Detailed monitoring state: #{result.instance_monitorings[0].monitoring.state}"
  return true
rescue Aws::EC2::Errors::InvalidState
  puts "The instance is not in a monitorable state. Skipping this step."
  return false
rescue StandardError => e
  puts "Error enabling detailed monitoring: #{e.message}"
  return false
end

# Displays information about available
# Amazon Elastic Compute Cloud (Amazon EC2) instances.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_instances_information(Aws::EC2::Client.new(region: 'us-east-1'))
def list_instances_information(ec2_client)
  result = ec2_client.describe_instances
  result.reservations.each do |reservation|
    if reservation.instances.count.positive?
      reservation.instances.each do |instance|
        puts '-' * 12
        puts "Instance ID:              #{instance.instance_id}"
```

```
        puts "State:                      #{instance.state.name}"
        puts "Image ID:                   #{instance.image_id}"
        puts "Instance type:              #{instance.instance_type}"
        puts "Architecture:               #{instance.architecture}"
        puts "IAM instance profile ARN:  #{instance.iam_instance_profile.arn}"
        puts "Key name:                   #{instance.key_name}"
        puts "Launch time:                #{instance.launch_time}"
        puts "Detailed monitoring state: #{instance.monitoring.state}"
        puts "Public IP address:          #{instance.public_ip_address}"
        puts "Public DNS name:            #{instance.public_dns_name}"
        puts "VPC ID:                     #{instance.vpc_id}"
        puts "Subnet ID:                  #{instance.subnet_id}"
        if instance.tags.count.positive?
          puts 'Tags:'
          instance.tags.each do |tag|
            puts "                            #{tag.key}/#{tag.value}"
          end
        end
      end
    end
  end
end

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-manage-instances.rb ' \
      'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-manage-instances.rb ' \
      'i-033c48ef067af3dEX us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-033c48ef067af3dEX'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end
```

```
  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'Attempting to stop the instance. ' \
    'This might take a few minutes...'
  unless instance_stopped?(ec2_client, instance_id)
    puts 'Cannot stop the instance. Skipping this step.'
  end

  puts "\nAttempting to restart the instance. " \
    'This might take a few minutes...'
  unless instance_restarted?(ec2_client, instance_id)
    puts 'Cannot restart the instance. Skipping this step.'
  end

  puts "\nAttempting to reboot the instance. " \
    'This might take a few minutes...'
  unless instance_rebooted?(ec2_client, instance_id)
    puts 'Cannot reboot the instance. Skipping this step.'
  end

  puts "\nAttempting to enable detailed monitoring for the instance..."
  unless instance_detailed_monitoring_enabled?(ec2_client, instance_id)
    puts 'Cannot enable detailed monitoring for the instance. ' \
      'Skipping this step.'
  end

  puts "\nInformation about available instances:"
  list_instances_information(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

Terminación de una instancia de Amazon EC2

En el siguiente ejemplo se intenta terminar una instancia de Amazon EC2 especificada.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Attempts to terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
```

```
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
    response.instance_statuses[0].instance_state.name == 'terminated'

    puts 'The instance is already terminated.'
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
  return true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  return false
end

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
```

```
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  unless instance_terminated?(ec2_client, instance_id)
    puts 'Could not terminate instance.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Obtención de información acerca de las regiones y las zonas de disponibilidad para Amazon EC2

El siguiente ejemplo:

1. Muestra una lista de las Regiones de AWS para Amazon EC2 que están disponibles para usted.
2. Muestra una lista de las zonas de disponibilidad de Amazon EC2 disponibles en función de la Región de AWS del cliente de Amazon EC2.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-ec2'

# Displays a list of AWS Regions for Amazon Elastic Compute Cloud (Amazon EC2)
# that are available to you.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-east-1'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
```

```
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print "  Endpoint\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name.to_s
    print ' ' * (max_region_string_length - region.region_name.length)
    print '  '
    print region.endpoint.to_s
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-east-1'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print '  Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print "  State\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_zone_string_length
  print '  '
```

```
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print ' ' * (max_region_string_length - zone.region_name.length)
    print '  '
    print zone.zone_name
    print ' ' * (max_zone_string_length - zone.zone_name.length)
    print '  '
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts '  Messages for this zone:'
      zone.messages.each do |message|
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end

# Full example call:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'AWS Regions for Amazon EC2 that are available to you:'
  list_regions_endpoints(ec2_client)
```

```
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS Region
 '#{region}':"
  list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

Ejemplos de AWS Elastic Beanstalk con AWS SDK para Ruby

AWS Elastic Beanstalk le permite implementar rápidamente y administrar aplicaciones en la nube de AWS sin tener que preocuparse por la infraestructura que las ejecuta. Puede utilizar los siguientes ejemplos para acceder a Elastic Beanstalk mediante AWS SDK para Ruby. Para obtener más información acerca de Elastic Beanstalk, consulte la documentación de AWS Elastic Beanstalk.

Temas

- Obtención de información acerca de todas las aplicaciones de AWS Elastic Beanstalk
- Obtención de información acerca de una aplicación específica de AWS Elastic Beanstalk
- Actualización de una aplicación Ruby on Rails para AWS Elastic Beanstalk

Obtención de información acerca de todas las aplicaciones de AWS Elastic Beanstalk

En el siguiente ejemplo se muestran los nombres, las descripciones y las direcciones URL de todas las aplicaciones de Elastic Beanstalk de la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-elasticbeanstalk'  # v2: require 'aws-sdk'

eb = Aws::ElasticBeanstalk::Client.new(region: 'us-west-2')
```

```
eb.describe_applications.applications.each do |a|
  puts "Name:         #{a.application_name}"
  puts "Description:  #{a.description}"

  eb.describe_environments({application_name: a.application_name}).environments.each do
 |env|
    puts "  Environment:  #{env.environment_name}"
    puts "    URL:        #{env.cname}"
    puts "    Health:     #{env.health}"
  end
end
```

Obtención de información acerca de una aplicación específica de AWS Elastic Beanstalk

En el siguiente ejemplo se muestran el nombre, la descripción y la URL de la aplicación `MyRailsApp` en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-elasticbeanstalk'  # v2: require 'aws-sdk'

eb = Aws::ElasticBeanstalk::Client.new(region: 'us-west-2')

app = eb.describe_applications({application_names: [args[0]]})

if app.exists?
  puts "Name:         #{app.application_name}"
  puts "Description:  #{app.description}"

  envs = eb.describe_environments({application_name: app.application_name})
  puts "URL:          #{envs.environments[0].cname}"
```

```
end
```

Actualización de una aplicación Ruby on Rails para AWS Elastic Beanstalk

En el siguiente ejemplo se actualiza la aplicación Ruby on Rails `MyRailsApp` en la región `us-west-2`.

> **Note**
>
> Debe situarse en la raíz de la aplicación Rails para ejecutar correctamente el script.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-elasticbeanstalk'  # v2: require 'aws-sdk'

Aws.config.update({region: 'us-west-2'})

eb = Aws::ElasticBeanstalk::Client.new
s3 = Aws::S3::Client.new

app_name = 'MyRailsApp'

# Get S3 bucket containing app
app_versions = eb.describe_application_versions({ application_name: app_name })
av = app_versions.application_versions[0]
bucket = av.source_bundle.s3_bucket
s3_key = av.source_bundle.s3_key

# Get info on environment
envs = eb.describe_environments({ application_name: app_name })
env = envs.environments[0]
```

```
env_name = env.environment_name

# Create new storage location
resp = eb.create_storage_location()

puts "Created storage location in bucket #{resp.s3_bucket}"

s3.list_objects({
  prefix: s3_key,
  bucket: bucket
})

# Create ZIP file
zip_file_basename = SecureRandom.urlsafe_base64.to_s
zip_file_name = zip_file_basename + '.zip'

# Call out to OS to produce ZIP file
cmd = "git archive --format=zip -o #{zip_file_name} HEAD"
%x[ #{cmd} ]

# Get ZIP file contents
zip_contents = File.read(zip_file_name)

key = app_name + "\\" + zip_file_name

s3.put_object({
  body: zip_contents,
  bucket: bucket,
  key: key
})

date = Time.new
today = date.day.to_s + "/" + date.month.to_s + "/" + date.year.to_s

eb.create_application_version({
  process: false,
  application_name: app_name,
  version_label: zip_file_basename,
  source_bundle: {
    s3_bucket: bucket,
    s3_key: key
  },
  description: "Updated #{today}"
})
```

```
eb.update_environment({
  environment_name: env_name,
  version_label: zip_file_basename
})
```

Ejemplos de AWS Identity and Access Management (IAM) con AWS SDK para Ruby

AWS Identity and Access Management (IAM) es un servicio web para controlar el acceso de forma segura a los servicios de Servicios de AWS. Puede utilizar los siguientes ejemplos para acceder a IAM mediante AWS SDK para Ruby. Para obtener más información acerca de IAM, consulte la documentación de IAM.

Temas

- Obtención de información acerca de los usuarios de IAM
- Obtención de una lista de usuarios de IAM que son administradores
- Adición de un usuario de IAM nuevo
- Creación de claves de acceso de usuario para un usuario de IAM
- Adición de una política administrada a un usuario de IAM
- Creación de un rol de IAM
- Administración de usuarios de IAM
- Trabajo con políticas de IAM
- Administración de las claves de acceso de IAM
- Trabajo con certificados de servidores de IAM
- Administración de alias de cuenta de IAM

Obtención de información acerca de los usuarios de IAM

En el siguiente ejemplo se muestra una lista de los grupos, las políticas y los ID de clave de acceso de todos los usuarios de IAM en la región `us-west-2`. Si hay más de 100 usuarios, `iam.list_users.IsTruncated` es true e `iam.list_users.Marker` contiene un valor que puede utilizar para obtener información sobre usuarios adicionales. Consulte el tema Aws::IAM::Client.list_users para obtener más información.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-iam'

# Displays information about available users in
# AWS Identity and Access Management (IAM) including users'
# names, associated group names, inline embedded user policy names,
# and access key IDs.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @example
#   get_user_details(Aws::IAM::Client.new)
def get_user_details(iam_client)
  users_response = iam_client.list_users

  if users_response.key?('users') && users_response.users.count.positive?

    # Are there more users available than can be displayed?
    if users_response.key?('is_truncated') && users_response.is_truncated
      puts '(Note: not all users are displayed here, ' \
        "only the first #{users_response.users.count}.)"
    else
      puts "Found #{users_response.users.count} user(s):"
    end

    users_response.users.each do |user|
      name = user.user_name
      puts '-' * 30
      puts "User name: #{name}"

      puts "Groups:"
      groups_response = iam_client.list_groups_for_user(user_name: name)
      if groups_response.key?('groups') &&
        groups_response.groups.count.positive?

        groups_response.groups.each do |group|
          puts "  #{group.group_name}"
        end
      else
        puts '  None'
      end
```

```
      puts 'Inline embedded user policies:'
      policies_response = iam_client.list_user_policies(user_name: name)
      if policies_response.key?('policy_names') &&
        policies_response.policy_names.count.positive?

        policies_response.policy_names.each do |policy_name|
          puts "  #{policy_name}"
        end
      else
        puts '  None'
      end

      puts 'Access keys:'
      access_keys_response = iam_client.list_access_keys(user_name: name)

      if access_keys_response.key?('access_key_metadata') &&
        access_keys_response.access_key_metadata.count.positive?

        access_keys_response.access_key_metadata.each do |access_key|
          puts "  #{access_key.access_key_id}"
        end
      else
        puts '  None'
      end
    end
  else
    puts 'No users found.'
  end
rescue StandardError => e
  puts "Error getting user details: #{e.message}"
end

# Full example call:
def run_me
  iam_client = Aws::IAM::Client.new
  puts 'Attempting to get details for available users...'
  get_user_details(iam_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

Obtención de una lista de usuarios de IAM que son administradores

En el siguiente ejemplo se utiliza el método get_account_authorization_details, para obtener la lista de usuarios para la cuenta actual.

Elija Copy para guardar el código localmente.

Cree el archivo get_admins.rb.

Añada la gema de IAM necesaria y la gema de os, y utilice esta última para utilizar el certificado agrupado si usa Microsoft Windows.

> **Note**
>
> La versión 2 de AWS SDK para Ruby no disponía de gemas específicas de los servicios.

```
require 'aws-sdk-iam'  # v2: require 'aws-sdk'
require 'os'

if OS.windows?
 Aws.use_bundled_cert!
end
```

Cree un método para determinar si el usuario tiene una política con privilegios de administrador.

```
def user_has_admin_policy(user, admin_access)
  policies = user.user_policy_list

  policies.each do |p|
    if p.policy_name == admin_access
      return true
    end
  end

  false
end
```

Cree un método para determinar si el usuario tiene una política asociada con privilegios de administrador.

```
def user_has_attached_policy(user, admin_access)
```

```
  attached_policies = user.attached_managed_policies

  attached_policies.each do |p|
    if p.policy_name == admin_access
      return true
    end
  end

  false
end
```

Cree un método para determinar si un grupo al que pertenece el usuario tiene una política con privilegios de administrador.

Cree un método para determinar si un grupo al que pertenece el usuario tiene una política asociada con privilegios de administrador.

```
def group_has_admin_policy(client, group, admin_access)
 resp = client.list_group_policies(
   group_name: group.group_name
 )

 resp.policy_names.each do |name|
   if name == admin_access
     return true
   end
 end

 false
end
```

Cree un método para determinar si un grupo al que pertenece el usuario tiene privilegios de administrador.

```
def user_has_admin_from_group(client, user, admin_access)
 resp = client.list_groups_for_user(
   user_name: user.user_name
 )

 resp.groups.each do |group|
   has_admin_policy = group_has_admin_policy(client, group, admin_access)
```

```
    if has_admin_policy
      return true
    end

    has_attached_policy = group_has_attached_policy(client, group, admin_access)
    if has_attached_policy
      return true
    end
  end

  false
end
```

Cree un método para determinar si el usuario tiene privilegios de administrador.

```
def is_user_admin(client, user, admin_access)
  has_admin_policy = user_has_admin_policy(user, admin_access)
  if has_admin_policy
    return true
  end

  has_attached_admin_policy = user_has_attached_policy(user, admin_access)
  if has_attached_admin_policy
    return true
  end

  has_admin_from_group = user_has_admin_from_group(client, user, admin_access)
  if has_admin_from_group
    return true
  end

  false
end
```

Cree un método que se ejecutará en bucle a través de una lista de usuarios y obtener la cantidad de dichos usuarios que tienen privilegios de administrador.

```
<code>
```

La rutina principal comienza aquí. Cree un cliente de IAM y variables para almacenar el número de usuarios, el número de usuarios que tienen privilegios de administrador y la cadena que identifica una política que suministra privilegios de administrador.

```
def get_admin_count(client, users, admin_access)
 num_admins = 0

 users.each do |user|
   is_admin = is_user_admin(client, user, admin_access)
   if is_admin
     puts user.user_name
     num_admins += 1
   end
 end

 num_admins
end
```

Llame a `get_account_authorization_details` para obtener los detalles de la cuenta y obtener los usuarios de la cuenta de `user_detail_list`. Realice un seguimiento de la cantidad obtenida de usuarios, llame a `get_admin_count` para obtener el número de usuarios que tienen privilegios de administrador, y realice un seguimiento del número de estos.

```
details = client.get_account_authorization_details(
 filter: ['User']
)

users = details.user_detail_list
num_users += users.count
more_admins = get_admin_count(client, users, access_admin)
num_admins += more_admins
```

Si en la primera llamada a `get_account_authorization_details` no se obtuvieron todos los detalles, llame de nuevo y repita el proceso para determinar cuántos tienen privilegios de administrador.

```
<code>
```

Por último, muestre cuántos usuarios tienen privilegios de administrador.

```

```

more_users = details.is_truncated

```
while more_users
    details = client.get_account_authorization_details (
        filter: [‘User’], marker: details.marker
    )
    users = details.user_detail_list
    num_users += users.count more_admins = get_admin_count(client, users, access_admin)
    num_admins += more_admins
    more_users = details.is_truncated

end
```

Consulte el ejemplo completo en GitHub.

Adición de un usuario de IAM nuevo

En el siguiente ejemplo se crea el `my_groovy_user` de usuario de IAM en la región `us-west-2` con el `REPLACE_ME` de contraseña y se muestra el ID de la cuenta del usuario. Si ya existe un usuario con ese nombre, se muestra un mensaje y no se crea un nuevo usuario.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-iam'

# Creates a user in AWS Identity and Access Management (IAM).
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @param initial_password [String] The initial password for the user.
# @return [String] The ID of the user if the user was created, otherwise;
#   the string 'Error'.
# @example
#   puts create_user(Aws::IAM::Client.new, 'my-user', 'my-!p@55w0rd!')
def create_user(iam_client, user_name, initial_password)
  response = iam_client.create_user(user_name: user_name)
  iam_client.wait_until(:user_exists, user_name: user_name)
  iam_client.create_login_profile(
    password: initial_password,
```

```
    password_reset_required: true,
    user_name: user_name
  )
  return response.user.user_id
rescue Aws::IAM::Errors::EntityAlreadyExists
  puts "Error creating user '#{user_name}': user already exists."
  return 'Error'
rescue StandardError => e
  puts "Error creating user '#{user_name}': #{e.message}"
  return 'Error'
end

# Full example call:
def run_me
  user_name = 'my-user'
  initial_password = 'my-!p@55w0rd!'
  iam_client = Aws::IAM::Client.new

  puts "Attempting to create user '#{user_name}'..."
  user_id = create_user(iam_client, user_name, initial_password)

  if user_id == 'Error'
    puts 'User not created.'
  else
    puts "User '#{user_name}' created with ID '#{user_id}' and initial " \
      "sign-in password '#{initial_password}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Creación de claves de acceso de usuario para un usuario de IAM

En el siguiente ejemplo se crea una clave de acceso y una clave secreta para el usuario de IAM `my_groovy_user` en la región `us-west-2`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-iam'

# Creates an access key for a user in AWS Identity and Access Management (IAM).
#
```

```
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @example
#   create_access_key(Aws::IAM::Client.new, 'my-user')
def create_access_key(iam, user_name)
  response = iam.create_access_key(user_name: user_name)
  access_key = response.access_key
  puts 'Access key created:'
  puts "  Access key ID: #{access_key.access_key_id}"
  puts "  Secret access key: #{access_key.secret_access_key}"
  puts 'Keep a record of this information in a secure location. ' \
    'This will be the only time you will be able to view the ' \
    'secret access key.'
rescue Aws::IAM::Errors::LimitExceeded
  puts 'Error creating access key: limit exceeded. Cannot create any more. ' \
    'To create more, delete an existing access key, and then try again.'
rescue StandardError => e
  puts "Error creating access key: #{e.message}"
end

# Full example call:
def run_me
  iam = Aws::IAM::Client.new
  user_name = 'my-user'

  puts 'Attempting to create an access key...'
  create_access_key(iam, user_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

Adición de una política administrada a un usuario de IAM

En el siguiente ejemplo se añade la política administrada `AmazonS3FullAccess` al usuario de IAM `my_groovy_user` en la región `us-west-2`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-iam'
```

```
# Attaches a policy to a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @param policy_arn [String] The Amazon Resource Name (ARN) of the policy.
# @return [Boolean] true if the policy was attached; otherwise, false.
# @example
#   exit 1 unless alias_created?(
#     Aws::IAM::Client.new,
#     'my-user',
#     'arn:aws:iam::aws:policy/AmazonS3FullAccess'
#   )
def policy_attached_to_user?(iam_client, user_name, policy_arn)
  iam_client.attach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
  return true
rescue StandardError => e
  puts "Error attaching policy to user: #{e.message}"
  return false
end

# Full example call:
def run_me
  user_name = 'my-user'
  arn_prefix = 'arn:aws:iam::aws:policy/'
  policy_arn = arn_prefix + 'AmazonS3FullAccess'
  iam_client = Aws::IAM::Client.new

  puts "Attempting to attach policy with ARN '#{policy_arn}' to " \
    "user '#{user_name}'..."

  if policy_attached_to_user?(iam_client, user_name, policy_arn)
    puts 'Policy attached.'
  else
    puts 'Policy not attached.'
  end
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

Creación de un rol de IAM

En el siguiente ejemplo se crea el rol `my_groovy_role` para que Amazon EC2 pueda acceder a Amazon S3 y a Amazon DynamoDB en la región `us-west-2`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

require 'aws-sdk-iam'

# Creates a role in AWS Access and Identity Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] A name for the role.
# @param assume_role_policy_document [String]
# @param policy_arns [Array] An array of type String representing
#   Amazon Resource Names (ARNs) corresponding to available
#   IAM managed policies.
# @return [String] The ARN of the new role; otherwise, the string 'Error'.
# @example
#   puts create_role(
#     Aws::IAM::Client.new,
#     'my-ec2-s3-dynamodb-full-access-role',
#     {
#       Version: '2012-10-17',
#       Statement: [
#         {
#           Effect: 'Allow',
#           Principal: {
#             Service: 'ec2.amazonaws.com'
#           },
#           Action: 'sts:AssumeRole'
#         }
#       ]
#     },
#     [
#       'arn:aws:iam::aws:policy/AmazonS3FullAccess',
#       'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess'
#     ]
#   )
def create_role(
```

```
  iam_client,
  role_name,
  assume_role_policy_document,
  policy_arns
)
  iam_client.create_role(
    role_name: role_name,
    assume_role_policy_document: assume_role_policy_document.to_json
  )
  policy_arns.each do |policy_arn|
    iam_client.attach_role_policy(
      policy_arn: policy_arn,
      role_name: role_name,
    )
  end
  return iam_client.get_role(role_name: role_name).role.arn
rescue StandardError => e
  puts "Error creating role: #{e.message}"
  return 'Error'
end

# Full example call:
def run_me
  role_name = 'my-ec2-s3-dynamodb-full-access-role'

  # Allow the role to trust Amazon Elastic Compute Cloud (Amazon EC2)
  # within the AWS account.
  assume_role_policy_document = {
    Version: '2012-10-17',
    Statement: [
      {
        Effect: 'Allow',
        Principal: {
          Service: 'ec2.amazonaws.com'
        },
        Action: 'sts:AssumeRole'
      }
    ]
  }

  # Allow the role to take all actions within
  # Amazon Simple Storage Service (Amazon S3)
  # and Amazon DynamoDB across the AWS account.
  policy_arns = [
```

```
    'arn:aws:iam::aws:policy/AmazonS3FullAccess',
    'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess'
  ]

  iam_client = Aws::IAM::Client.new

  puts "Attempting to create the role named '#{role_name}'..."

  role_arn = create_role(
    iam_client,
    role_name,
    assume_role_policy_document,
    policy_arns
  )

  if role_arn == 'Error'
    puts 'Could not create role.'
  else
    puts "Role created with ARN '#{role_arn}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Administración de usuarios de IAM

Un usuario de IAM representa a la persona o el servicio que interactúa con AWS. Para obtener más información acerca de los usuarios de IAM, consulte Usuarios de IAM.

En este ejemplo, se utiliza AWS SDK para Ruby con IAM para:

1. Obtener información acerca de los usuarios de AWS IAM disponibles mediante Aws::IAM::Client#list_users.
2. Crear un usuario mediante Aws::IAM::Client#create_user.
3. Actualizar el nombre del usuario mediante Aws::IAM::Client#update_user.
4. Eliminar el usuario mediante Aws::IAM::Client#delete_user.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

Ejemplo

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

# The following code example shows how to to:
# 1. Get a list of user names in AWS Identity and Access Management (IAM).
# 2. Create a user.
# 3. Update the user's name.
# 4. Delete the user.

require 'aws-sdk-iam'

# Gets a list of available user names in
# AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @example
#   list_user_names(Aws::IAM::Client.new)
def list_user_names(iam_client)
  response = iam_client.list_users
  if response.key?('users') && response.users.count.positive?
    response.users.each do |user|
      puts user.user_name
    end
  else
    puts 'No users found.'
  end
rescue StandardError => e
  puts "Error listing user names: #{e.message}"
end

# Creates a user in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the new user.
# @return [Boolean] true if the user was created; otherwise, false.
# @example
#   exit 1 unless user_created?(Aws::IAM::Client.new, 'my-user')
```

```
def user_created?(iam_client, user_name)
  iam_client.create_user(user_name: user_name)
  return true
rescue Aws::IAM::Errors::EntityAlreadyExists
  puts "Error creating user: user '#{user_name}' already exists."
  return false
rescue StandardError => e
  puts "Error creating user: #{e.message}"
  return false
end

# Changes the name of a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_current_name [String] The current name of the user.
# @param user_new_name [String] The new name for the user.
# @return [Boolean] true if the name of the user was changed;
#   otherwise, false.
# @example
#   exit 1 unless user_name_changed?(
#     Aws::IAM::Client.new,
#     'my-user',
#     'my-changed-user'
#   )
def user_name_changed?(iam_client, user_current_name, user_new_name)
  iam_client.update_user(
    user_name: user_current_name,
    new_user_name: user_new_name
  )
  return true
rescue StandardError => e
  puts "Error updating user name: #{e.message}"
  return false
end

# Deletes a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
```

```
# @param user_name [String] The name of the user.
# @return [Boolean] true if the user was deleted; otherwise, false.
# @example
#   exit 1 unless user_deleted?(Aws::IAM::Client.new, 'my-user')
def user_deleted?(iam_client, user_name)
  iam_client.delete_user(user_name: user_name)
  return true
rescue StandardError => e
  puts "Error deleting user: #{e.message}"
  return false
end

# Full example call:
def run_me
  user_name = 'my-user'
  user_changed_name = 'my-changed-user'
  delete_user = true
  iam_client = Aws::IAM::Client.new

  puts "Initial user names are:\n\n"
  list_user_names(iam_client)

  puts "\nAttempting to create user '#{user_name}'..."

  if user_created?(iam_client, user_name)
    puts 'User created.'
  else
    puts 'Could not create user. Stopping program.'
    exit 1
  end

  puts "User names now are:\n\n"
  list_user_names(iam_client)

  puts "\nAttempting to change the name of the user '#{user_name}' " \
    "to '#{user_changed_name}'..."

  if user_name_changed?(iam_client, user_name, user_changed_name)
    puts 'User name changed.'
    puts "User names now are:\n\n"
    list_user_names(iam_client)

    if delete_user
      # Delete user with changed name.
```

```
      puts "\nAttempting to delete user '#{user_changed_name}'..."

      if user_deleted?(iam_client, user_changed_name)
        puts 'User deleted.'
      else
        puts 'Could not delete user. You must delete the user yourself.'
      end

      puts "User names now are:\n\n"
      list_user_names(iam_client)
    end
  else
    puts 'Could not change user name.'
    puts "User names now are:\n\n"
    list_user_names(iam_client)

    if delete_user
      # Delete user with initial name.
      puts "\nAttempting to delete user '#{user_name}'..."

      if user_deleted?(iam_client, user_name)
        puts 'User deleted.'
      else
        puts 'Could not delete user. You must delete the user yourself.'
      end

      puts "User names now are:\n\n"
      list_user_names(iam_client)
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Trabajo con políticas de IAM

Una política de IAM es un documento que especifica uno o varios permisos. Para obtener más información sobre las políticas de IAM, consulte Descripción general de políticas de IAM.

En este ejemplo, se utiliza AWS SDK para Ruby con IAM para:

1. Crear una política utilizando Aws::IAM::Client#create_policy.
2. Obtener información sobre la política utilizando Aws::IAM::Client#get_policy.

3. Adjuntar la política a un rol usando Aws::IAM::Client#attach_role_policy.
4. Mostrar las políticas adjuntas al rol usando Aws::IAM::Client#list_attached_role_policies.
5. Separar la política del rol usando Aws::IAM::Client#detach_role_policy.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

También es necesario crear el rol (my-role) especificado en el script. Puede hacerlo en la consola de IAM.

Ejemplo

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

# The following code example shows how to:
# 1. Create a policy in AWS Identity and Access Management (IAM).
# 2. Attach the policy to a role.
# 3. List the policies that are attached to the role.
# 4. Detach the policy from the role.

require 'aws-sdk-iam'

# Creates a policy in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param policy_name [String] A name for the policy.
# @param policy_document [Hash] The policy definition.
# @return [String] The new policy's Amazon Resource Name (ARN);
#   otherwise, the string 'Error'.
# @example
#   puts create_policy(
#     Aws::IAM::Client.new,
#     'my-policy',
#     {
#       'Version': '2012-10-17',
```

```
#       'Statement': [
#         {
#           'Effect': 'Allow',
#           'Action': 's3:ListAllMyBuckets',
#           'Resource': 'arn:aws:s3:::*'
#         }
#       ]
#     }
#   )
def create_policy(iam_client, policy_name, policy_document)
  response = iam_client.create_policy(
    policy_name: policy_name,
    policy_document: policy_document.to_json
  )
  return response.policy.arn
rescue StandardError => e
  puts "Error creating policy: #{e.message}"
  return 'Error'
end

# Attaches a policy to a role in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - An existing role.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to attach the policy to.
# @param policy_arn [String] The policy's Amazon Resource Name (ARN).
# @return [Boolean] True if the policy was attached to the role;
#   otherwise, false.
# @example
#   exit 1 unless policy_attached_to_role?(
#     Aws::IAM::Client.new,
#     'my-role',
#     'arn:aws:iam::111111111111:policy/my-policy'
#   )
def policy_attached_to_role?(iam_client, role_name, policy_arn)
  iam_client.attach_role_policy(role_name: role_name, policy_arn: policy_arn)
  return true
rescue StandardError => e
  puts "Error attaching policy to role: #{e.message}"
  return false
end
```

```
# Displays a list of policy Amazon Resource Names (ARNs) that are attached to a
# role in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - An existing role.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role.
# @example
#   list_policy_arns_attached_to_role(Aws::IAM::Client.new, 'my-role')
def list_policy_arns_attached_to_role(iam_client, role_name)
  response = iam_client.list_attached_role_policies(role_name: role_name)
  if response.key?('attached_policies') && response.attached_policies.count.positive?
    response.attached_policies.each do |attached_policy|
      puts "  #{attached_policy.policy_arn}"
    end
  else
    puts 'No policies attached to role.'
  end
rescue StandardError => e
  puts "Error checking for policies attached to role: #{e.message}"
end

# Detaches a policy from a role in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - An existing role with an attached policy.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to detach the policy from.
# @param policy_arn [String] The policy's Amazon Resource Name (ARN).
# @return [Boolean] True if the policy was detached from the role;
#   otherwise, false.
# @example
#   exit 1 unless policy_detached_from_role?(
#     Aws::IAM::Client.new,
#     'my-role',
#     'arn:aws:iam::111111111111:policy/my-policy'
#   )
def policy_detached_from_role?(iam_client, role_name, policy_arn)
  iam_client.detach_role_policy(role_name: role_name, policy_arn: policy_arn)
  return true
rescue StandardError => e
  puts "Error detaching policy from role: #{e.message}"
```

```
  return false
end

# Full example call:
def run_me
  role_name = 'my-role'
  policy_name = 'my-policy'

  # Allows the caller to get a list of all buckets in
  # Amazon Simple Storage Service (Amazon S3) that are owned by the caller.
  policy_document = {
    'Version': '2012-10-17',
    'Statement': [
      {
        'Effect': 'Allow',
        'Action': 's3:ListAllMyBuckets',
        'Resource': 'arn:aws:s3:::*'
      }
    ]
  }

  detach_policy_from_role = true
  iam_client = Aws::IAM::Client.new

  puts "Attempting to create policy '#{policy_name}'..."
  policy_arn = create_policy(iam_client, policy_name, policy_document)

  if policy_arn == 'Error'
    puts 'Could not create policy. Stopping program.'
    exit 1
  else
    puts 'Policy created.'
  end

  puts "Attempting to attach policy '#{policy_name}' " \
    "to role '#{role_name}'..."

  if policy_attached_to_role?(iam_client, role_name, policy_arn)
    puts 'Policy attached.'
  else
    puts 'Could not attach policy to role.'
    detach_policy_from_role = false
  end
```

```
  puts "Policy ARNs attached to role '#{role_name}':"
  list_policy_arns_attached_to_role(iam_client, role_name)

  if detach_policy_from_role
    puts "Attempting to detach policy '#{policy_name}' " \
      "from role '#{role_name}'..."

    if policy_detached_from_role?(iam_client, role_name, policy_arn)
      puts 'Policy detached.'
    else
      puts 'Could not detach policy from role. You must detach it yourself.'
    end

  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Administración de las claves de acceso de IAM

Los usuarios necesitan sus propias claves de acceso para realizar llamadas programáticas AWS desde AWS SDK para Ruby. Para atender esta necesidad, puede crear, modificar, ver o rotar claves de acceso (ID de clave de acceso y claves de acceso secretas) de los usuarios de IAM. De forma predeterminada, cuando se crea una clave de acceso, su estado es activo. Esto significa que el usuario puede utilizar la clave de acceso para realizar llamadas a la API. Para obtener más información acerca de las claves de acceso, consulte Administración de claves de acceso de usuarios de IAM.

En este ejemplo, se utiliza AWS SDK para Ruby con IAM para:

1. Obtener una lista de las claves de acceso de los usuarios de AWS IAM mediante Aws::IAM::Client#list_access_keys.
2. Crear una clave de acceso mediante Aws::IAM::Client#create_access_key.
3. Determinar cuándo se han utilizado por última vez las claves de acceso mediante Aws::IAM::Client#get_access_key_last_used.
4. Desactivar las claves de acceso mediante Aws::IAM::Client#update_access_key.
5. Eliminar la clave de acceso mediante Aws::IAM::Client#delete_access_key.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

También debe crear el usuario (my-user) especificado en el script. Puede crear un nuevo usuario de IAM en la consola de IAM o de forma programada, tal y como se muestra en Adding a New IAM User (Añadir un nuevo usuario de IAM).

Ejemplo

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

# This code example demonstrates how to:
# 1. List access keys for a user in AWS Identity and Access Management (IAM).
# 2. Create an access key for a user.
# 3. Determine when a user's access keys were last used.
# 4. Deactivate an access key for a user.
# 5. Delete an access key for a user.

require 'aws-sdk-iam'

# Lists information about access keys for a user in
# AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @example
#   puts list_access_keys(Aws::IAM::Client.new, 'my-user')
def list_access_keys(iam, user_name)
  response = iam.list_access_keys(user_name: user_name)

  if response.access_key_metadata.count.positive?
    puts 'Access key IDs:'
    response.access_key_metadata.each do |key_metadata|
```

```
      puts "  #{key_metadata.access_key_id}"
    end
  else
    puts "No access keys found for user '#{user_name}'."
  end
rescue Aws::IAM::Errors::NoSuchEntity
  puts "Error listing access keys: cannot find user '#{user_name}'."
  exit 1
rescue StandardError => e
  puts "Error listing access keys: #{e.message}"
end

# Creates an access key for a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @return [Aws::IAM::Types::AccessKey] Information about the new access key;
#   otherwise, the string 'Error'.
# @example
#   puts create_access_key(Aws::IAM::Client.new, 'my-user')
def create_access_key(iam, user_name)
  response = iam.create_access_key(user_name: user_name)
  access_key = response.access_key
  puts 'Access key created:'
  puts "  Access key ID: #{access_key.access_key_id}"
  puts "  Secret access key: #{access_key.secret_access_key}"
  puts 'Keep a record of this information in a secure location. ' \
    'This will be the only time you will be able to view the ' \
    'secret access key.'
  return access_key
rescue Aws::IAM::Errors::LimitExceeded
  puts 'Error creating access key: limit exceeded. Cannot create any more. ' \
    'To create more, delete an existing access key, and then try again.'
  return 'Error'
rescue StandardError => e
  puts "Error creating access key: #{e.message}"
  return 'Error'
end

# Lists information about when access keys for a user in
# AWS Identity and Access Management (IAM) were last used.
```

```
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @example
#   puts access_keys_last_used(Aws::IAM::Client.new, 'my-user')
def access_keys_last_used(iam, user_name)
  response = iam.list_access_keys(user_name: user_name)

  response.access_key_metadata.each do |key_metadata|
    last_used = iam.get_access_key_last_used(access_key_id: key_metadata.access_key_id)
    if last_used.access_key_last_used.last_used_date.nil?
      puts "  Key '#{key_metadata.access_key_id}' not used or date undetermined."
    else
      puts "  Key '#{key_metadata.access_key_id}' last used on " \
      "#{last_used.access_key_last_used.last_used_date}"
    end
  end
rescue StandardError => e
  puts "Error determining when access keys were last used: #{e.message}"
end

# Deactivates an access key in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - A user in IAM.
# - An access key for that user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID of the access key.
# @return [Boolean] true if the access key was deactivated;
#   otherwise, false.
# @example
#   exit 1 unless access_key_deactivated?(
#     Aws::IAM::Client.new,
#     'my-user',
#     'AKIAIOSFODNN7EXAMPLE'
#   )
def access_key_deactivated?(iam, user_name, access_key_id)
  iam.update_access_key(
    user_name: user_name,
```

```
    access_key_id: access_key_id,
    status: 'Inactive'
  )
  return true
rescue StandardError => e
  puts "Error deactivating access key: #{e.message}"
  return false
end

# Deletes an access key in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - A user in IAM.
# - An access key for that user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID of the access key.
# @return [Boolean] true if the access key was deleted;
#   otherwise, false.
# @example
#   exit 1 unless access_key_deleted?(
#     Aws::IAM::Client.new,
#     'my-user',
#     'AKIAIOSFODNN7EXAMPLE'
#   )
def access_key_deleted?(iam, user_name, access_key_id)
  iam.delete_access_key(
    user_name: user_name,
    access_key_id: access_key_id
  )
  return true
rescue StandardError => e
  puts "Error deleting access key: #{e.message}"
  return false
end

# Full example call:
def run_me
  iam = Aws::IAM::Client.new
  user_name = 'my-user'
  create_key = true  # Set to false to not create a new access key.
  delete_key = true  # Set to false to not delete any generated access key.
```

```
  puts "Access keys for user '#{user_name}' before attempting to create an " \
    'additional access key for the user:'
  list_access_keys(iam, user_name)

  access_key = ''

  if create_key
    puts 'Attempting to create an additional access key...'
    access_key = create_access_key(iam, user_name)

    if access_key == 'Error'
      puts 'Additional access key not created. Stopping program.'
      exit 1
    end

    puts 'Additional access key created. Access keys for user now are:'
    list_access_keys(iam, user_name)
  end

  puts 'Determining when current access keys were last used...'
  access_keys_last_used(iam, user_name)

  if create_key && delete_key
    puts 'Attempting to deactivate additional access key...'

    if access_key_deactivated?(iam, user_name, access_key.access_key_id)
      puts 'Access key deactivated. Access keys for user now are:'
      list_access_keys(iam, user_name)
    else
      puts 'Access key not deactivated. Stopping program.'
      puts 'You will need to delete the access key yourself.'
    end

    puts 'Attempting to delete additional access key...'

    if access_key_deleted?(iam, user_name, access_key.access_key_id)
      puts 'Access key deleted. Access keys for user now are:'
      list_access_keys(iam, user_name)
    else
      puts 'Access key not deleted. You will need to delete the ' \
        'access key yourself.'
    end
  end
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

Trabajo con certificados de servidores de IAM

Para habilitar las conexiones HTTPS en su sitio web o aplicación en AWS, necesita un certificado de servidor SSL/TLS. Para utilizar un certificado obtenido de un proveedor externo en su sitio web o aplicación en AWS, debe cargar el certificado en IAM o importarlo a AWS Certificate Manager. Para obtener más información acerca de los certificados de servidor, consulte Uso de certificados de servidor.

En este ejemplo, se utiliza AWS SDK para Ruby con IAM para:

1. Actualizar un certificado de servidor mediante Aws::IAM::Client#update_server_certificate.
2. Eliminar el certificado de servidor mediante Aws::IAM::Client#delete_server_certificate.
3. Mostrar información acerca de los certificados de servidor restante mediante Aws::IAM::Client#list_server_certificates.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

> **Note**
>
> El certificado de servidor ya debe existir, de lo contrario el script producirá un error Aws::IAM::Errores::NoSuchEntity.

Ejemplo

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

# The following code example shows how to:
# 1. Update a server certificate in AWS Identity and Access Management (IAM).
```

```
# 2. List the names of available server certificates.
# 3. Delete a server certificate.

require 'aws-sdk-iam'

# Gets a list of available server certificate names in
# AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @example
#   list_server_certificate_names(Aws::IAM::Client.new)
def list_server_certificate_names(iam_client)
  response = iam_client.list_server_certificates

  if response.key?('server_certificate_metadata_list') &&
    response.server_certificate_metadata_list.count.positive?

    response.server_certificate_metadata_list.each do |certificate_metadata|
      puts certificate_metadata.server_certificate_name
    end
  else
    puts 'No server certificates found. Stopping program.'
    exit 1
  end
rescue StandardError => e
  puts "Error getting server certificate names: #{e.message}"
end

# Changes the name of a server certificate in
# AWS Identity and Access Management (IAM).
#
# Prerequisites:
#
# - The server certificate in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param server_certificate_current_name [String] The current name of
#   the server certificate.
# @param server_certificate_new_name [String] The new name for the
#   the server certificate.
# @return [Boolean] true if the name of the server certificate
#   was changed; otherwise, false.
# @example
#   exit 1 unless server_certificate_name_changed?(
```

```
#     Aws::IAM::Client.new,
#     'my-server-certificate',
#     'my-changed-server-certificate'
#   )
def server_certificate_name_changed?(
  iam_client,
  server_certificate_current_name,
  server_certificate_new_name
)
  iam_client.update_server_certificate(
    server_certificate_name: server_certificate_current_name,
    new_server_certificate_name: server_certificate_new_name
  )
  return true
rescue StandardError => e
  puts "Error updating server certificate name: #{e.message}"
  return false
end

# Deletes a server certificate in
# AWS Identity and Access Management (IAM).
#
# Prerequisites:
#
# - The server certificate in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param server_certificate_name [String] The name of the server certificate.
# @return [Boolean] true if the server certificate was deleted;
#   otherwise, false.
# @example
#   exit 1 unless server certificate_deleted?(
#     Aws::IAM::Client.new,
#     'my-server-certificate'
#   )
def server_certificate_deleted?(iam_client, server_certificate_name)
  iam_client.delete_server_certificate(
    server_certificate_name: server_certificate_name
  )
  return true
rescue StandardError => e
  puts "Error deleting server certificate: #{e.message}"
  return false
end
```

```
# Full example call:
def run_me
  server_certificate_name = 'my-server-certificate'
  server_certificate_changed_name = 'my-changed-server-certificate'
  delete_server_certificate = true
  iam_client = Aws::IAM::Client.new

  puts "Initial server certificate names are:\n\n"
  list_server_certificate_names(iam_client)

  puts "\nAttempting to change name of server certificate " \
    " '#{server_certificate_name}' " \
    "to '#{server_certificate_changed_name}'..."

  if server_certificate_name_changed?(
    iam_client,
    server_certificate_name,
    server_certificate_changed_name
  )
    puts 'Server certificate name changed.'
    puts "Server certificate names now are:\n\n"
    list_server_certificate_names(iam_client)

    if delete_server_certificate
      # Delete server certificate with changed name.
      puts "\nAttempting to delete server certificate " \
        "'#{server_certificate_changed_name}'..."

      if server_certificate_deleted?(iam_client, server_certificate_changed_name)
        puts 'Server certificate deleted.'
      else
        puts 'Could not delete server certificate. You must delete it yourself.'
      end

      puts "Server certificate names now are:\n\n"
      list_server_certificate_names(iam_client)
    end
  else
    puts 'Could not change server certificate name.'
    puts "Server certificate names now are:\n\n"
    list_server_certificate_names(iam_client)

    if delete_server_certificate
```

```
      # Delete server certificate with initial name.
      puts "\nAttempting to delete server certificate '#{server_certificate_name}'..."

      if server_certificate_deleted?(iam_client, server_certificate_name)
        puts 'Server certificate deleted.'
      else
        puts 'Could not delete server certificate. You must delete it yourself.'
      end

      puts "Server certificate names now are:\n\n"
      list_server_certificate_names(iam_client)
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Administración de alias de cuenta de IAM

Si quiere que la dirección URL de la página de inicio de sesión contenga el nombre de su empresa u otro identificador intuitivo en lugar de su ID de cuenta de AWS, puede crear un alias de cuenta de IAM para el ID de cuenta de AWS. Si crea un alias de cuenta de IAM, la dirección URL de su página de inicio de sesión cambia para incorporar dicho alias. Para obtener más información sobre los alias de cuenta, consulte Su ID de cuenta y alias de AWS.

En este ejemplo, se utiliza AWS SDK para Ruby con IAM para:

1. Mostrar una lista de alias de la cuenta de AWS, mediante Aws::IAM::Client#list_account_aliases.
2. Crear un alias de cuenta, utilizando Aws::IAM::Client#create_account_alias.
3. Eliminar el alias de cuenta, utilizando Aws::IAM::Client#delete_account_alias.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

En el código de ejemplo, cambie la cadena my-account-alias por algo único para todos los productos de Amazon Web Services.

Ejemplo

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX - License - Identifier: Apache - 2.0

# The following code example shows how to:
# 1. List available AWS account aliases.
# 2. Create an account alias.
# 3. Delete an account alias.

require 'aws-sdk-iam'

# Lists available AWS account aliases.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @example
#   puts list_aliases(Aws::IAM::Client.new)
def list_aliases(iam)
  response = iam.list_account_aliases

  if response.account_aliases.count.positive?
    response.account_aliases.each do |account_alias|
      puts "  #{account_alias}"
    end
  else
    puts 'No account aliases found.'
  end
rescue StandardError => e
  puts "Error listing account aliases: #{e.message}"
end

# Creates an AWS account alias.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param account_alias [String] The name of the account alias to create.
# @return [Boolean] true if the account alias was created; otherwise, false.
# @example
#   exit 1 unless alias_created?(Aws::IAM::Client.new, 'my-account-alias')
def alias_created?(iam, account_alias)
  iam.create_account_alias(account_alias: account_alias)
  return true
```

```
rescue StandardError => e
  puts "Error creating account alias: #{e.message}"
  return false
end

# Deletes an AWS account alias.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param account_alias [String] The name of the account alias to delete.
# @return [Boolean] true if the account alias was deleted; otherwise, false.
# @example
#   exit 1 unless alias_deleted?(Aws::IAM::Client.new, 'my-account-alias')
def alias_deleted?(iam, account_alias)
  iam.delete_account_alias(account_alias: account_alias)
  return true
rescue StandardError => e
  puts "Error deleting account alias: #{e.message}"
  return false
end

# Full example call:
def run_me
  iam = Aws::IAM::Client.new
  account_alias = 'my-account-alias'
  create_alias = true # Change to false to not generate an account alias.
  delete_alias = true # Change to false to not delete any generated account alias.

  puts 'Account aliases are:'
  list_aliases(iam)

  if create_alias
    puts 'Attempting to create account alias...'
    if alias_created?(iam, account_alias)
      puts 'Account alias created. Account aliases now are:'
      list_aliases(iam)
    else
      puts 'Account alias not created. Stopping program.'
      exit 1
    end
  end

  if create_alias && delete_alias
    puts 'Attempting to delete account alias...'
    if alias_deleted?(iam, account_alias)
```

```
      puts 'Account alias deleted. Account aliases now are:'
      list_aliases(iam)
    else
      puts 'Account alias not deleted. You will need to delete ' \
        'the alias yourself.'
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Ejemplos de AWS Key Management Service con AWS SDK para Ruby

AWS Key Management Service (AWS KMS) es un servicio de cifrado y administración de claves escalado para la nube. Puede utilizar los siguientes ejemplos para acceder a AWS KMS mediante AWS SDK para Ruby. Para obtener más información acerca de AWS KMS, consulte la documentación de AWS KMS. Para obtener más información acerca de cliente de AWS KMS, consulte Aws::KMS::Client.

Temas

- Creación de un AWS KMS key
- Cifrado de datos en AWS KMS
- Descifrado de un blob de datos en AWS KMS
- Recifrado de un blob de datos en AWS KMS

Creación de un AWS KMS key

En el siguiente ejemplo se utiliza el método create_key de AWS SDK for Ruby, que implementa la operación CreateKey para crear una AWS KMS keys. Dado que el ejemplo solo cifra una pequeña cantidad de datos, una clave de KMS está bien para nuestros propios fines. Para una mayor cantidad de datos, utilice la clave de KMS para cifrar una clave de cifrado de datos (DEK).

```
require "aws-sdk-kms" # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
```

```
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: "CreatedBy",
                               tag_value: "ExampleUser"
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

Consulte el ejemplo completo en GitHub.

Cifrado de datos en AWS KMS

En el siguiente ejemplo se utiliza el método de cifrado de AWS SDK para Ruby, que implementa la operación de cifrado, para cifrar la cadena “1234567890”. El ejemplo muestra una versión inteligible del blob de cifrado resultante.

```
require "aws-sdk-kms"  # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

text = "1234567890"

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.encrypt({
  key_id: keyId,
  plaintext: text,
})

# Display a readable version of the resulting encrypted blob.
puts "Blob:"
puts resp.ciphertext_blob.unpack("H*")
```

Consulte el ejemplo completo en GitHub.

Descifrado de un blob de datos en AWS KMS

En el siguiente ejemplo se utiliza el método de cifrado de AWS SDK para Ruby que implementa la operación de cifrado, para descifrar la cadena proporcionada y emitir el resultado.

```
require "aws-sdk-kms" # v2: require 'aws-sdk'

# Decrypted blob

blob =
 "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596e09
blob_packed = [blob].pack("H*")

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts "Raw text: "
puts resp.plaintext
```

Consulte el ejemplo completo en GitHub.

Recifrado de un blob de datos en AWS KMS

En el siguiente ejemplo se utiliza el método re_encrypt de AWS SDK for Ruby, que implementa la operación ReEncrypt, para descifrar los datos cifrados y volver a cifrar inmediatamente los datos con una nueva AWS KMS key. Las operaciones se realizan en su totalidad en el servidor en AWS KMS, por lo que su texto no cifrado nunca se expondrá fuera de AWS KMS. El ejemplo muestra una versión inteligible del blob que se ha vuelto a cifrar resultante.

```
require "aws-sdk-kms"  # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
 "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596e09
```

```
sourceCiphertextBlob = [blob].pack("H*")

# Replace the fictitious key ARN with a valid key ID

destinationKeyId = "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.re_encrypt({
  ciphertext_blob: sourceCiphertextBlob,
  destination_key_id: destinationKeyId
})

# Display a readable version of the resulting re-encrypted blob.
puts "Blob:"
puts resp.ciphertext_blob.unpack("H*")
```

Consulte el ejemplo completo en GitHub.

Ejemplos de AWS Lambda con AWS SDK para Ruby

AWS Lambda (Lambda) es una plataforma de computación para desarrolladores web backend que no requiere ninguna labor de administración y ejecuta el código en la nube de AWS y le proporciona una estructura de precios detallada. Puede utilizar los siguientes ejemplos para acceder a Lambda mediante AWS SDK para Ruby. Para obtener más información acerca de Lambda, consulte la documentación de AWS Lambda.

Temas

- Visualización de información acerca de todas las funciones de Lambda
- Creación de una función de Lambda
- Ejecución de una función de Lambda
- Configuración de una función de Lambda para recibir notificaciones

Visualización de información acerca de todas las funciones de Lambda

En el siguiente ejemplo se muestran el nombre, el ARN y el rol de todas las funciones de Lambda en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-lambda'  # v2: require 'aws-sdk'

client = Aws::Lambda::Client.new(region: 'us-west-2')

client.list_functions.functions.each do |function|
  puts 'Name: ' + function.function_name
  puts 'ARN:  ' + function.function_arn
  puts 'Role: ' + function.role
  puts
end
```

Creación de una función de Lambda

En el siguiente ejemplo se crea la función de Lambda denominada `my-notification-function` en la región `us-west-2` mediante estos valores:

- ARN de rol: `my-resource-arn`. En la mayoría de los casos, solo debe adjuntar la política administrada `AWSLambdaExecute` a la política de este rol.
- Punto de entrada de la función: `my-package.my-class`
- Tiempo de ejecución: `java8`
- Archivo zip: `my-zip-file.zip`
- Bucket: `my-notification-bucket`
- Clave: `my-zip-file`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
```

```
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-lambda'  # v2: require 'aws-sdk'

client = Aws::Lambda::Client.new(region: 'us-west-2')

args = {}
args[:role] = 'my-resource-arn'
args[:function_name] = 'my-notification-function'
args[:handler] = 'my-package.my-class'

# Also accepts nodejs, nodejs4.3, and python2.7
args[:runtime] = 'java8'

code = {}
code[:zip_file] = 'my-zip-file.zip'
code[:s3_bucket] = 'my-notification-bucket'
code[:s3_key] = 'my-zip-file'

args[:code] = code

client.create_function(args)
```

Ejecución de una función de Lambda

El siguiente ejemplo ejecuta la función de Lambda denominada `MyGetitemsFunction` en la región `us-west-2`. Esta función devuelve una lista de elementos de una base de datos. La JSON de entrada sería similar al siguiente.

```
{
   "SortBy": "name|time",
   "SortOrder": "ascending|descending",
   "Number": 50
}
```

donde:

- `SortBy` son los criterios para ordenar los resultados. En nuestros ejemplos se utiliza `time`, lo que significa que los elementos devueltos se ordenan en el orden en que se han añadido a la base de datos.
- `SortOrder` es el criterio de ordenación. En nuestro ejemplo se utiliza `descending`, lo que significa que el elemento más reciente es el último de la lista.
- `Number` es el número máximo de elementos que se recuperan (el valor predeterminado es 50). En nuestro ejemplo se utiliza `10`, lo que significa que se obtienen los 10 elementos más recientes.

El JSON de salida tiene el siguiente aspecto, donde:

- `STATUS-CODE` es un código de estado HTTP, `200` significa que la llamada se ha realizado correctamente.
- `RESULT` es el resultado de la llamada, ya sea `success` o `failure`.
- `ERROR` es un mensaje de error si `result` es `failure`, de lo contrario se devuelve una cadena vacía.
- `DATA` es una matriz de los resultados que se devuelven si `result` es `success`, de lo contrario se devuelve el valor nil.

```
{
   "statusCode": "STATUS-CODE",
   "body": {
      "result": "RESULT",
      "error": "ERROR",
      "data": "DATA"
   }
}
```

El primer paso consiste en cargar los módulos que utilizamos:

- `aws-sdk` carga el módulo de AWS SDK para Ruby que utilizamos para invocar la función de Lambda.
- `json` carga el módulo JSON que utilizamos para activar y desactivar la serialización de las cargas de la solicitud y la respuesta.
- `os` carga el módulo de sistema operativo que utilizamos para garantizar que podemos ejecutar la aplicación de Ruby en Microsoft Windows. Si utiliza otro sistema operativo, puede quitar estas líneas.

- A continuación, cree el cliente de Lambda, que utilizamos para invocar la función de Lambda.
- A continuación, cree el hash para los argumentos de la solicitud y llame a `MyGetItemsFunction`.
- Por último, analice la respuesta y, si son correctos, imprima los elementos.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-lambda'  # v2: require 'aws-sdk'
require 'json'

# To run on Windows:
require 'os'
if OS.windows?
  Aws.use_bundled_cert!
end

client = Aws::Lambda::Client.new(region: 'us-west-2')

# Get the 10 most recent items
req_payload = {:SortBy => 'time', :SortOrder => 'descending', :NumberToGet => 10}
payload = JSON.generate(req_payload)

resp = client.invoke({
                       function_name: 'MyGetItemsFunction',
                       invocation_type: 'RequestResponse',
                       log_type: 'None',
                       payload: payload
                     })

resp_payload = JSON.parse(resp.payload.string) # , symbolize_names: true)

# If the status code is 200, the call succeeded
if resp_payload["statusCode"] == 200
```

```
  # If the result is success, we got our items
  if resp_payload["body"]["result"] == "success"
    # Print out items
    resp_payload["body"]["data"].each do |item|
      puts item
    end
  end
end
```

Consulte el ejemplo completo en GitHub.

Configuración de una función de Lambda para recibir notificaciones

En el siguiente ejemplo se configura la función de Lambda denominada `my-notification-function` en la región `us-west-2` para aceptar notificaciones procedentes del recurso con el ARN `my-resource-arn`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-lambda'  # v2: require 'aws-sdk'

client = Aws::Lambda::Client.new(region: 'us-west-2')

args = {}
args[:function_name] = 'my-notification-function'
args[:statement_id] = 'lambda_s3_notification'
args[:action] = 'lambda:InvokeFunction'
args[:principal] = 's3.amazonaws.com'
args[:source_arn] = 'my-resource-arn'

client.add_permission(args)
```

Ejemplos de Amazon Polly con AWS SDK para Ruby

Amazon Polly es un servicio en la nube que convierte el texto en un segmento hablado muy realista. Los ejemplos de AWS SDK para Ruby pueden integrar Amazon Polly en sus aplicaciones. Obtenga más información sobre Amazon Polly en la documentación de Amazon Polly. En los ejemplos, se presupone que ya ha creado y configurado el SDK (es decir, ha importado todos los paquetes necesarios y establecido sus credenciales y su región). Para obtener más información, consulte Instalación de AWSSDK para Ruby y Configuración de AWS SDK para Ruby.

Temas

Obtención de una lista de voces

En este ejemplo se utiliza el método describe_voices para obtener la lista de las voces del idioma inglés de EE. UU. en la región `us-west-2`.

Elija `Copy` para guardar el código localmente.

Cree el archivo polly_describe_voices.rb.

Añada la gema necesaria.

> **Note**
>
> La versión 2 de AWS SDK para Ruby no disponía de gemas específicas de los servicios.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
```

```
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-polly'  # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts '  ' + v.gender
    puts
  end
rescue StandardError => ex
  puts 'Could not get voices'
  puts 'Error message:'
  puts ex.message
end
```

Consulte el ejemplo completo en GitHub.

Obtención de una lista de lexicones

En este ejemplo se utiliza el método list_lexicons para obtener la lista de lexicones en la región `us-west-2`.

Elija `Copy` para guardar el código localmente.

Cree el archivo polly_list_lexicons.rb.

Añada la gema necesaria.

> **Note**
>
> La versión 2 de AWS SDK para Ruby no disponía de gemas específicas de los servicios.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-polly'  # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts '  Alphabet:' + l.attributes.alphabet
    puts '  Language:' + l.attributes.language
    puts
  end
rescue StandardError => ex
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts ex.message
end
```

Consulte el ejemplo completo en GitHub.

Sintetizar el habla

En este ejemplo se utiliza el método synthesize_speech para obtener el texto desde un archivo y generar un archivo MP3 que contiene la voz sintetizada.

Elija Copy para guardar el código localmente.

Cree el archivo polly_synthesize_speech.rb.

Añada la gema necesaria.

> **Note**
>
> zLa versión 2 de AWS SDK para Ruby no disponía de gemas específicas de los servicios.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-polly'  # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?()
    puts 'You must supply a filename'
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts 'No such file: ' + filename
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
```

```
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
    output_format: "mp3",
    text: contents,
    voice_id: "Joanna",
  })

  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = first_part + '.mp3'

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts 'Wrote MP3 content to: ' + mp3_file
rescue StandardError => ex
  puts 'Got error:'
  puts 'Error message:'
  puts ex.message
end
```

> **Note**
>
> El archivo MP3 resultante está en formato MPEG-2.

Consulte el ejemplo completo en GitHub.

Ejemplos de Amazon RDS con AWS SDK para Ruby

Amazon Relational Database Service (Amazon RDS) es un servicio web que facilita la configuración, la operación y la escala de una base de datos relacional en la nube. Puede utilizar los siguientes ejemplos para acceder a Amazon RDS mediante AWS SDK para Ruby. Para obtener más información sobre Amazon RDS, consulte la documentación sobre Amazon Relational Database Service.

> **Note**
>
> Algunos de los siguientes ejemplos utilizan métodos que se introdujeron en la versión `2.2.18` de la clase `Aws::RDS::Resource`. Para ejecutar esos ejemplos, debe utilizar esa versión o una posterior de la gema `aws-sdk`.

Temas

Obtención de información acerca de todas las instancias de Amazon RDS

En el siguiente ejemplo se muestran el nombre (ID) y el estado de sus instancias de Amazon RDS en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')
```

```
rds.db_instances.each do |i|
  puts "Name (ID): #{i.id}"
  puts "Status   : #{i.db_instance_status}"
  puts
end
```

Obtención de información acerca de todas las instantáneas de Amazon RDS

En el siguiente ejemplo se muestran los nombres (ID) y el estado de todas las instantáneas (instancias) de Amazon RDS en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_snapshots.each do |s|
  puts "Name (ID): #{s.snapshot_id}"
  puts "Status:    #{s.status}"
end
```

Obtención de información sobre todos los clústeres de Amazon RDS y sus instantáneas

En el siguiente ejemplo se muestra el nombre (ID) y el estado de todos los clústeres de Amazon RDS y el nombre (ID) y el estado de sus instantáneas en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
```

```
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_clusters.each do |c|
  puts "Name (ID): #{c.id}"
  puts "Status:    #{c.status}"

  c.snapshots.each do |s|
    puts "  Snapshot: #{s.snapshot_id}"
    puts "  Status:   #{s.status}"
  end
end
```

Obtención de información acerca de todos los grupos de seguridad de Amazon RDS

En el siguiente ejemplo se muestra una lista con los nombres de todos los grupos de seguridad de Amazon RDS en la región `us-west-2`.

> **Note**
>
> Los grupos de seguridad de Amazon RDS son de aplicación cuando se utiliza la plataforma de Amazon EC2-Classic, únicamente. Si utiliza Amazon EC2-VPC, utilice los grupos de seguridad de VPC. Ambos se muestran en el ejemplo.

> **Warning**
>
> Vamos a retirar EC2-Classic el 15 de agosto de 2022. Le recomendamos que migre de EC2-Classic a una VPC. Para obtener más información, consulte Migración de EC2-Classic a una VPC en la Guía del usuario de Amazon EC2 para instancias de Linux o en la Guía del usuario de Amazon EC2 para instancias de Windows. Consulte también la entrada de blog

EC2-Classic Networking is Retiring – Here's How to Prepare (Se va a retirar la red EC2-Classic: cómo prepararse).

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_instances.each do |i|
  # Show any security group IDs and descriptions
  puts 'Security Groups:'

  i.db_security_groups.each do |sg|
    puts sg.db_security_group_name
    puts '  ' + sg.db_security_group_description
    puts
  end

  # Show any VPC security group IDs and their status
  puts 'VPC Security Groups:'

  i.vpc_security_groups.each do |vsg|
    puts vsg.vpc_security_group_id
    puts '  ' + vsg.status
    puts
  end
end
```

Obtención de información acerca de todos los grupos de subredes de Amazon RDS

En el siguiente ejemplo se muestran el nombre y el estado de todos los grupos de subredes de Amazon RDS en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_subnet_groups.each do |s|
  puts s.name
  puts '  ' + s.subnet_group_status
end
```

Obtención de información acerca de todos los grupos de parámetros de Amazon RDS

En el siguiente ejemplo se muestra una lista con los nombres y las descripciones de todos los grupos de parámetros de Amazon RDS de la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.
```

```
require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_parameter_groups.each do |p|
  puts p.db_parameter_group_name
  puts '  ' + p.description
end
```

Creación de instantáneas de instancias de Amazon RDS

En el siguiente ejemplo se crea una instantánea de la instancia de Amazon RDS representada por instance_name en la región `us-west-2`.

> **Note**
>
> Si la instancia es miembro de un clúster, no podrá crear una instantánea de la instancia. En su lugar, debe crear una instantánea del clúster (consulte Crear una instantánea de un clúster de Amazon RDS).

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

instance = rds.db_instance(instance_name)

date =  Time.new
date_time =  date.year.to_s +  '-' +  date.month.to_s +  '-' +  date.day.to_s +  '-' +
 date.hour.to_s +  '-' +  date.min.to_s
```

```
id = instance_name + '-' + date_time

instance.create_snapshot({db_snapshot_identifier: id})

puts "Created snapshot #{id}"
```

Creación de instantáneas de clústeres de Amazon RDS

En el siguiente ejemplo se crea una instantánea del clúster de Amazon RDS representado por cluster_name en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

cluster = rds.db_cluster(cluster_name)

date =  Time.new
date_time =  date.year.to_s +  '-' +  date.month.to_s +  '-' +  date.day.to_s +  '-' +
 date.hour.to_s +  '-' +  date.min.to_s

id = cluster_name + '-' + date_time

cluster.create_snapshot({db_cluster_snapshot_identifier: id})

puts "Created cluster snapshot #{id}"
```

Ejemplos de Amazon SES con AWS SDK para Ruby

Amazon Simple Email Service (Amazon SES) es una plataforma de correo electrónico que ofrece un método sencillo y rentable de enviar y recibir correo electrónico a través de los propios dominios y direcciones de correo electrónico. Puede utilizar los siguientes ejemplos para acceder a Amazon SES mediante AWS SDK para Ruby. Para obtener más información acerca de Amazon SES, consulte la documentación de Amazon SES.

Temas

Obtención de una lista de direcciones de correo electrónico de Amazon SES válidas

El siguiente ejemplo ilustra cómo utilizar AWS SDK for Ruby para obtener una lista de las direcciones de correo electrónico de Amazon SES válidas.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-ses'  # v2: require 'aws-sdk'

# Create client in us-west-2 region
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})
```

```
ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

Consulte el ejemplo completo en GitHub.

Verificación de direcciones de correo electrónico en Amazon SES

El siguiente ejemplo ilustra cómo utilizar AWS SDK for Ruby para verificar una dirección de correo electrónico de Amazon SES.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-ses'  # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = "recipient@example.com"

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
```

```
begin
  ses.verify_email_identity({
    email_address: recipient
  })

  puts 'Email sent to ' + recipient

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"
end
```

Consulte el ejemplo completo en GitHub.

Envío de un mensaje a una dirección de correo electrónico en Amazon SES

El siguiente ejemplo ilustra cómo utilizar AWS SDK para Ruby para enviar un mensaje a una dirección de correo electrónico de Amazon SES.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-ses'  # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
```

```
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
```

```
        data: subject
      }
    },
    source: sender,
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts 'Email sent to ' + recipient


# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"
end
```

Consulte el ejemplo completo en GitHub.

Obtención de las estadísticas de Amazon SES

El siguiente ejemplo ilustra cómo utilizar AWS SDK for Ruby para obtener estadísticas sobre Amazon SES. Utilice esta información para evitar dañar su reputación cuando los mensajes de correo electrónico rebotan o se rechazan.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-ses'  # v2: require 'aws-sdk'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

begin
```

```
  # Get send statistics so we don't ruin our reputation
  resp = ses.get_send_statistics({})

  dps = resp.send_data_points

  puts "Got #{dps.count} data point(s):"
  puts

  dps.each do |dp|
    puts "Timestamp:  #{dp.timestamp}" #=> Time
    puts "Attempts:   #{dp.delivery_attempts}" #=> Integer
    puts "Bounces:    #{dp.bounces}" #=> Integer
    puts "Complaints: #{dp.complaints}" #=> Integer
    puts "Rejects:    #{dp.rejects}"  #-> Integer
    puts
  end

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Error: #{error}"
end
```

Consulte el ejemplo completo en GitHub.

Ejemplos de Amazon SNS con AWS SDK para Ruby

Amazon Simple Notification Service (Amazon SNS) es un servicio web que permite a las aplicaciones, los usuarios finales y los dispositivos enviar y recibir notificaciones de forma instantánea desde la nube. Puede utilizar los siguientes ejemplos para acceder a Amazon SNS mediante AWS SDK para Ruby. Para obtener más información acerca de Amazon SNS, consulte la documentación de Amazon SNS.

Temas

- Obtención de información acerca de todos los temas de Amazon SNS
- Creación de un tema de Amazon SNS
- Obtención de información acerca de todas las suscripciones de un tema de Amazon SNS
- Creación de una suscripción en un tema de Amazon SNS
- Envío de un mensaje a todos los suscriptores de un tema de Amazon SNS
- Habilitación de un recurso para publicarlo en un tema de Amazon SNS

Obtención de información acerca de todos los temas de Amazon SNS

En el siguiente ejemplo se muestran los ARN de los temas de Amazon SNS de la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

sns.topics.each do |topic|
  puts topic.arn
end
```

Creación de un tema de Amazon SNS

En el siguiente ejemplo se crea el tema `MyGroovyTopic` en la región `us-west-2` y se muestra el ARN del tema resultante.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.
```

```
require 'aws-sdk-sns'  # v2: require 'aws-sdk'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.create_topic(name: 'MyGroovyTopic')
puts topic.arn
```

Obtención de información acerca de todas las suscripciones de un tema de Amazon SNS

En el siguiente ejemplo se muestran las direcciones de correo electrónico de las suscripciones de Amazon SNS para el tema con el ARN `arn:aws:sns:us-west-2:123456789:MyGroovyTopic` en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

topic.subscriptions.each do |s|
  puts s.attributes['Endpoint']
end
```

Creación de una suscripción en un tema de Amazon SNS

En el siguiente ejemplo se crea una suscripción para el tema con el ARN `arn:aws:sns:us-west-2:123456789:MyGroovyTopic` para un usuario que tiene la dirección de correo electrónico `MyGroovyUser@MyGroovy.com` en la región `us-west-2` y muestra el ARN resultante. Al

comienzo, el valor ARN está pendiente de confirmación. Cuando el usuario confirma la dirección de correo electrónico, este valor se convierte en un ARN real.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

sub = topic.subscribe({
  protocol: 'email',
  endpoint: 'MyGroovyUser@MyGroovy.com'
})

puts sub.arn
```

Envío de un mensaje a todos los suscriptores de un tema de Amazon SNS

En el siguiente ejemplo se envía el mensaje "Hello!" a todos los suscriptores del tema de Amazon SNS con el ARN `arn:aws:sns:us-west-2:123456789:MyGroovyTopic`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
```

```
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

topic.publish({
  message: 'Hello!'
})
```

Habilitación de un recurso para publicarlo en un tema de Amazon SNS

En el siguiente ejemplo se habilita el recurso con el ARN `my-resource-arn` para publicar en el tema con el ARN `my-topic-arn` en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

policy  = '{
  "Version":"2008-10-17",
  "Id":"__default_policy_ID",
  "Statement":[{
    "Sid":"__default_statement_ID",
    "Effect":"Allow",
    "Principal":{
      "AWS":"*"
    },
    "Action":["SNS:Publish"],
    "Resource":"' + my-topic-arn + '",
    "Condition":{
```

```
      "ArnEquals":{
        "AWS:SourceArn":"' + my-resource-arn + '"}
     }
  }]
}'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

# Get topic by ARN
topic = sns.topic(my-topic-arn)

# Add policy to topic
topic.set_attributes({
  attribute_name: "Policy",
  attribute_value: policy
})
```

Ejemplos de Amazon SQS con AWS SDK para Ruby

Amazon Simple Queue Service (Amazon SQS) es un servicio de colas de mensajes completamente administrado que facilita el desacople y el escalado de microservicios, sistemas distribuidos y aplicaciones sin servidor. Puede utilizar los siguientes ejemplos para acceder a Amazon SQS mediante AWS SDK para Ruby. Para obtener más información acerca de Amazon SQS, consulte la documentación de Amazon SQS.

Temas

- Obtención de información acerca de todas las colas de Amazon SQS
- Creación de una cola en Amazon SQS
- Trabajo con colas de Amazon SQS
- Envío de mensajes en Amazon SQS
- Envío y recepción de mensajes en Amazon SQS
- Recepción de mensajes en Amazon SQS
- Recepción de mensajes mediante el sondeo largo de Amazon SQS
- Habilitación del sondeo largo en Amazon SQS
- Recepción de mensajes utilizando la clase QueuePoller en Amazon SQS
- Redireccionamiento de mensajes fallidos en Amazon SQS
- Eliminación de una cola en Amazon SQS

- Habilitación de un recurso para publicarlo en una cola de Amazon SQS
- Trabajo con una cola de mensajes fallidos en Amazon SQS
- Especificación del tiempo de espera de visibilidad de los mensajes en Amazon SQS

Obtención de información acerca de todas las colas de Amazon SQS

En el siguiente ejemplo se muestran las URL, los ARN, los mensajes disponibles y los mensajes en tránsito de las colas de Amazon SQS en la región `us-west-2`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Lists the URLs of available queues in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-east-1'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
```

```
    attribute_names: [ "All" ]
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end

rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'

  sqs_client = Aws::SQS::Client.new(region: region)

  puts 'Listing available queue URLs...'
  list_queue_urls(sqs_client)

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
  queue_url = 'https://sqs.' + region + '.amazonaws.com/' +
    sts_client.get_caller_identity.account + '/' + queue_name

  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end

run_me if $PROGRAM_NAME == __FILE__
```

Creación de una cola en Amazon SQS

En el siguiente ejemplo se crea la cola de SQS denominada `MyGroovyQueue` en la región `us-west-2` y se muestra su URL.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

require 'aws-sdk-sqs'
```

```
# Creates a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
end

# Full example call:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts 'Queue created.'
  else
    puts 'Queue not created.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Trabajo con colas de Amazon SQS

Amazon SQS ofrece colas alojadas altamente escalables para almacenar los mensajes que se transfieren entre aplicaciones o microservicios. Para obtener más información acerca de las colas, consulte Funcionamiento de Amazon SQS.

En este ejemplo, se utiliza AWS SDK para Ruby con Amazon SQS para:

1. Obtener una lista de sus colas utilizando `Aws::SQS::Client#list_queues`.
2. Crear una cola mediante utilizando `Aws::SQS::Client#create_queue`.
3. Obtener la dirección URL de la cola utilizando `Aws::SQS::Client#get_queue_url`.
4. Eliminar la cola utilizando `Aws::SQS::Client#delete_queue`.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

Ejemplo

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to:
# 1. Get a list of your queues.
# 2. Create a queue.
# 3. Get the queue's URL.
# 4. Delete the queue.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Get a list of your queues.
sqs.list_queues.queue_urls.each do |queue_url|
  puts queue_url
```

```
end

# Create a queue.
queue_name = "my-queue"

begin
  sqs.create_queue({
    queue_name: queue_name,
    attributes: {
      "DelaySeconds" => "60", # Delay message delivery for 1 minute (60 seconds).
      "MessageRetentionPeriod" => "86400" # Delete message after 1 day (24 hours * 60
 minutes * 60 seconds).
    }
  })
rescue Aws::SQS::Errors::QueueDeletedRecently
  puts "A queue with the name '#{queue_name}' was recently deleted. Wait at least 60
 seconds and try again."
  exit(false)
end

# Get the queue's URL.
queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url
puts queue_url

# Delete the queue.
sqs.delete_queue(queue_url: queue_url)
```

Envío de mensajes en Amazon SQS

En el siguiente ejemplo se envía el mensaje "Hello world" a través de la cola de Amazon SQS con la URL URL en la región `us-west-2`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Sends a message to a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
```

```
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end

# Full example call:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  message_body = 'This is my message.'

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
  queue_url = 'https://sqs.' + region + '.amazonaws.com/' +
    sts_client.get_caller_identity.account + '/' + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending a message to the queue named '#{queue_name}'..."

  if message_sent?(sqs_client, queue_url, message_body)
    puts 'Message sent.'
  else
    puts 'Message not sent.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

En el siguiente ejemplo se envían los mensajes "Hello world" y "How is the weather?" a través de la cola de Amazon SQS con la dirección URL URL en la región `us-west-2`.

> **Note**
>
> Si la cola es una cola FIFO, debe incluir un parámetro `message_group_id` además de los parámetros `id` y `message_body`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Sends multiple messages as a batch to a queue in
# Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
```

```
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
  queue_url = 'https://sqs.' + region + '.amazonaws.com/' +
    sts_client.get_caller_identity.account + '/' + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Envío y recepción de mensajes en Amazon SQS

Después de crear una cola en Amazon SQA, puede enviarle un mensaje y después consumirlo. Para obtener más información, consulte Tutorial: Envío de un mensaje a una cola de Amazon SQS y Tutorial: Recepción y eliminación de un mensaje de una cola de Amazon SQS.

En este ejemplo, se utiliza AWS SDK para Ruby con Amazon SQS para:

1. Enviar un mensaje a una cola existente utilizando Aws::SQS::Client#send_message.

> **Note**
>
> Si la cola es una cola FIFO, debe incluir un parámetro `message_group_id` además de los parámetros `id` y `message_body`.

1. Recibir el mensaje en la cola de mensajes usando Aws::SQS::Client#receive_message.
2. Mostrar información sobre el mensaje.
3. Eliminar el mensaje de la cola de mensajes usando Aws::SQS::Client#delete_message.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

También es necesario crear la cola my-queue, lo que puede hacer en la consola de Amazon SQS.

Ejemplo

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
```

```
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to:
# 1. Send a message to a queue.
# 2. Receive the message in the queue.
# 3. Display information about the message.
# 4. Delete the message from the queue.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Send a message to a queue.
queue_name = "my-queue"

begin
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Create a message with three custom attributes: Title, Author, and WeeksOn.
  send_message_result = sqs.send_message({
    queue_url: queue_url,
    message_body: "Information about current NY Times fiction bestseller for week of
 2016-12-11.",
    message_attributes: {
      "Title" => {
        string_value: "The Whistler",
        data_type: "String"
      },
      "Author" => {
        string_value: "John Grisham",
        data_type: "String"
      },
      "WeeksOn" => {
        string_value: "6",
        data_type: "Number"
      }
    }
  })
rescue Aws::SQS::Errors::NonExistentQueue
  puts "A queue named '#{queue_name}' does not exist."
  exit(false)
```

```
end

puts send_message_result.message_id

# Receive the message in the queue.
receive_message_result = sqs.receive_message({
  queue_url: queue_url,
  message_attribute_names: ["All"], # Receive all custom attributes.
  max_number_of_messages: 1, # Receive at most one message.
  wait_time_seconds: 0 # Do not wait to check for the message.
})

# Display information about the message.
# Display the message's body and each custom attribute value.
receive_message_result.messages.each do |message|
  puts message.body
  puts "Title: #{message.message_attributes["Title"]["string_value"]}"
  puts "Author: #{message.message_attributes["Author"]["string_value"]}"
  puts "WeeksOn: #{message.message_attributes["WeeksOn"]["string_value"]}"

  # Delete the message from the queue.
  sqs.delete_message({
    queue_url: queue_url,
    receipt_handle: message.receipt_handle
  })
end
```

Recepción de mensajes en Amazon SQS

En el siguiente ejemplo se muestra el cuerpo de hasta 10 mensajes en la cola de Amazon SQS con la URL URL en la región `us-west-2`.

> **Note**
>
> `receive_message` no garantiza la obtención de todos los mensajes (consulte Propiedades de colas distribuidas) y de forma predeterminada no elimina el mensaje.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

require 'aws-sdk-sqs'
```

```
require 'aws-sdk-sts'

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue',
#     10
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)

  if max_number_of_messages > 10
    puts 'Maximum number of messages to receive must be 10 or less. ' \
      'Stopping program.'
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )

  if response.messages.count.zero?
    puts 'No messages to receive, or all messages have already ' \
      'been previously received.'
    return
  end

  response.messages.each do |message|
    puts '-' * 20
    puts "Message body: #{message.body}"
    puts "Message ID:   #{message.message_id}"
  end

rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
```

```
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
  queue_url = 'https://sqs.' + region + '.amazonaws.com/' +
    sts_client.get_caller_identity.account + '/' + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end

run_me if $PROGRAM_NAME == __FILE__
```

Recepción de mensajes mediante el sondeo largo de Amazon SQS

En el siguiente ejemplo se espera hasta 10 segundos para mostrar el cuerpo de hasta 10 mensajes en la cola de Amazon SQS con la URL URL en la región `us-west-2`.

Si no se especifica un tiempo de espera, el valor predeterminado es 0 (Amazon SQS no espera).

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-west-2')
```

```
resp = sqs.receive_message(queue_url: URL, max_number_of_messages: 10,
 wait_time_seconds: 10)

resp.messages.each do |m|
  puts m.body
end
```

Habilitación del sondeo largo en Amazon SQS

El sondeo largo ayuda a bajar el costo de usar Amazon SQS porque reduce el número de respuestas vacías y elimina las respuestas vacías falsas. Para obtener más información acerca del sondeo largo, consulte Sondeo largo de Amazon SQS.

En este ejemplo, se utiliza AWS SDK para Ruby con Amazon SQS para:

1. Crear una cola y establecerla para sondeo largo utilizando Aws::SQS::Client#create_queue.
2. Establecer el sondeo largo para una cola existente utilizando Aws::SQS::Client#set_queue_attributes.
3. Establecer el sondeo largo cuando se reciben mensajes en una cola utilizando Aws::SQS::Client#receive_message.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

También es necesario crear las colas existing-queue y receive-queue, lo que puede hacer en la consola de Amazon SQS.

Ejemplo

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
```

```
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to:
# 1. Create a queue and set it for long polling.
# 2. Set long polling for an existing queue.
# 3. Set long polling when receiving messages for a queue.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Create a queue and set it for long polling.
new_queue_name = "new-queue"

create_queue_result = sqs.create_queue({
  queue_name: new_queue_name,
  attributes: {
    "ReceiveMessageWaitTimeSeconds" => "20" # Wait 20 seconds to receive messages.
  },
})

puts create_queue_result.queue_url

# Set long polling for an existing queue.
begin
  existing_queue_name = "existing-queue"
  existing_queue_url = sqs.get_queue_url(queue_name: existing_queue_name).queue_url

  sqs.set_queue_attributes({
    queue_url: existing_queue_url,
    attributes: {
      "ReceiveMessageWaitTimeSeconds" => "20" # Wait 20 seconds to receive messages.
    },
  })
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot set long polling for a queue named '#{existing_queue_name}', as it does
 not exist."
end
```

```
# Set long polling when receiving messages for a queue.

# 1. Using receive_message.
begin
  receive_queue_name = "receive-queue"
  receive_queue_url = sqs.get_queue_url(queue_name: receive_queue_name).queue_url

  puts "Begin receipt of any messages using receive_message..."
  receive_message_result = sqs.receive_message({
    queue_url: receive_queue_url,
    attribute_names: ["All"], # Receive all available built-in message attributes.
    message_attribute_names: ["All"], # Receive any custom message attributes.
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Received #{receive_message_result.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages using receive_message for a queue named
 '#{receive_queue_name}', as it does not exist."
end

# 2. Using Aws::SQS::QueuePoller.
begin
  puts "Begin receipt of any messages using Aws::SQS::QueuePoller..."
  puts "(Will keep polling until no more messages available for at least 60 seconds.)"
  poller = Aws::SQS::QueuePoller.new(receive_queue_url)

  poller_stats = poller.poll({
    max_number_of_messages: 10,
    idle_timeout: 60 # Stop polling after 60 seconds of no more messages available
 (polls indefinitely by default).
  }) do |messages|
    messages.each do |message|
      puts "Message body: #{message.body}"
    end
  end
  # Note: If poller.poll is successful, all received messages are automatically deleted
 from the queue.

  puts "Poller stats:"
  puts "  Polling started at: #{poller_stats.polling_started_at}"
  puts "  Polling stopped at: #{poller_stats.polling_stopped_at}"
  puts "  Last message received at: #{poller_stats.last_message_received_at}"
  puts "  Number of polling requests: #{poller_stats.request_count}"
```

```
  puts "  Number of received messages: #{poller_stats.received_message_count}"
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages using Aws::SQS::QueuePoller for a queue named
 '#{receive_queue_name}', as it does not exist."
end
```

Recepción de mensajes utilizando la clase QueuePoller en Amazon SQS

En el siguiente ejemplo se utiliza la clase de utilidad `QueuePoller` para mostrar el cuerpo de todos los mensajes de la cola de Amazon SQS con la dirección URL URL en la región `us-west-2` y se elimina el mensaje. Después de 15 segundos de inactividad aproximadamente, el tiempo de espera del script finaliza.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

Aws.config.update({region: 'us-west-2'})

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(idle_timeout: 15) do |msg|
  puts msg.body
end
```

En el siguiente ejemplo se realiza un bucle en la cola de Amazon SQS con la URL URL y se espera hasta que trascurran los segundos de duración.

Puede obtener la dirección URL correcta ejecutando el ejemplo de Amazon SQS incluido en Obtener información acerca de todas las colas en Amazon SQS.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

Aws.config.update({region: 'us-west-2'})

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(wait_time_seconds: duration, idle_timeout: duration + 1) do |msg|
  puts msg.body
end
```

En el siguiente ejemplo se realiza un bucle en la cola de Amazon SQS con la dirección URL URL y se ofrecen los segundos del tiempo de espera de visibilidad para procesar el mensaje, representado por el método `do_something`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

# Process the message
def do_something(msg)
  puts msg.body
end
```

```
Aws.config.update({region: 'us-west-2'})

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(visibility_timeout: timeout, idle_timeout: timeout + 1) do |msg|
  do_something(msg)
end
```

En el siguiente ejemplo se realiza un bucle en la cola de Amazon SQS con la URL URL y se cambian los segundos del tiempo de espera de visibilidad para los mensajes que necesiten procesamiento adicional por el método do_something2.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

# Process the message
def do_something(_)
  true
end

# Do additional processing
def do_something2(msg)
  puts msg.body
end

Aws.config.update({region: 'us-west-2'})

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(idle_timeout: timeout + 1) do |msg|
  if do_something(msg)
```

```
    # need more time for processing
    poller.change_message_visibility_timeout(msg, timeout)

    do_something2(msg)
  end
end
```

Redireccionamiento de mensajes fallidos en Amazon SQS

En el siguiente ejemplo, todos los mensajes fallidos de la cola con la URL URL se redirigen a la cola con el ARN ARN.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.set_queue_attributes({
  queue_url: URL,
  attributes:
    {
      'RedrivePolicy' => "{\"maxReceiveCount\":\"5\", \"deadLetterTargetArn\":
\"#{ARN}\"}"
    }
})
```

Eliminación de una cola en Amazon SQS

En el siguiente ejemplo se elimina la cola de Amazon SQS con la URL URL en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
```

```
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

Habilitación de un recurso para publicarlo en una cola de Amazon SQS

En el siguiente ejemplo se habilita el recurso con el ARN `my-resource-arn` para publicarlo en la cola con el ARN `my-queue-arn` y la URL `my-queue-url` en la región `us-west-2`.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-west-2')

policy  = '{
  "Version":"2008-10-17",
  "Id":' + my-queue-arn + '/SQSDefaultPolicy",
  "Statement":[{
    "Sid":"__default_statement_ID",
    "Effect":"Allow",
```

```
    "Principal":{
      "AWS":"*"
    },
    "Action":["SQS:SendMessage"],
    "Resource":"' + my-queue-arn + '",
    "Condition":{
      "ArnEquals":{
        "AWS:SourceArn":"' + my-resource-arn + '"}
     }
  }]
}'

sqs.set_queue_attributes({
  queue_url: my-queue-url,
  attributes: {
    Policy: policy
  }
})
```

Trabajo con una cola de mensajes fallidos en Amazon SQS

Amazon SQS añade compatibilidad con las colas de mensajes fallidos. Una cola de mensajes fallidos es una cola a la que otras pueden enviar mensajes que no se pueden procesar correctamente. Puede apartar y aislar estos mensajes en la cola de mensajes fallidos para determinar por qué no se procesaron correctamente. Para obtener más información las colas de mensajes fallidos, consulte Using Amazon SQS Dead Letter Queues.

En este ejemplo, se utiliza AWS SDK para Ruby con Amazon SQS para:

1. Crear una cola que represente una cola de mensajes fallidos usando Aws::SQS::Client#create_queue.
2. Asociar la cola de mensajes fallidos a una cola existente utilizando Aws::SQS::Client#set_queue_attributes.
3. Enviar un mensaje a la cola existente utilizando Aws::SQS::Client#send_message.
4. Sondear la cola utilizando Aws::SQS::QueuePoller.
5. Recibir mensajes en la cola de mensajes fallidos usando Aws::SQS::Client#receive_message.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

Asimismo, necesitará utilizar la AWS Management Console para crear la cola my-queue.

> **Note**
>
> Para simplificar, en este código de ejemplo no se muestra Aws::SQS::Client#add_permission. En un escenario real, debe siempre restringir el acceso a acciones como SendMessage, ReceiveMessage, DeleteMessage y DeleteQueue. No hacerlo podría provocar la revelación de información, la denegación de servicio o la inyección de mensajes en las colas.

Ejemplo

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to:
# 1. Create a queue representing a dead letter queue.
# 2. Associate the dead letter queue with an existing queue.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

# Uncomment for Windows.
# Aws.use_bundled_cert!
```

```
sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Create a queue representing a dead letter queue.
dead_letter_queue_name = "dead-letter-queue"

sqs.create_queue({
  queue_name: dead_letter_queue_name
})

# Get the dead letter queue's URL and ARN, so that you can associate it with an
 existing queue.
dead_letter_queue_url = sqs.get_queue_url(queue_name: dead_letter_queue_name).queue_url

dead_letter_queue_arn = sqs.get_queue_attributes({
  queue_url: dead_letter_queue_url,
  attribute_names: ["QueueArn"]
}).attributes["QueueArn"]

# Associate the dead letter queue with an existing queue.
begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Use a redrive policy to specify the dead letter queue and its behavior.
  redrive_policy = {
    "maxReceiveCount" => "5", # After the queue receives the same message 5 times, send
 that message to the dead letter queue.
    "deadLetterTargetArn" => dead_letter_queue_arn
  }.to_json

  sqs.set_queue_attributes({
    queue_url: queue_url,
    attributes: {
      "RedrivePolicy" => redrive_policy
    }
  })

rescue Aws::SQS::Errors::NonExistentQueue
  puts "A queue named '#{queue_name}' does not exist."
  exit(false)
end

# Send a message to the queue.
puts "Sending a message..."
```

```
sqs.send_message({
  queue_url: queue_url,
  message_body: "I hope I get moved to the dead letter queue."
})

30.downto(0) do |i|
  print "\rWaiting #{i} second(s) for sent message to be receivable..."
  sleep(1)
end

puts "\n"

poller = Aws::SQS::QueuePoller.new(queue_url)
# Receive 5 messages max and stop polling after 20 seconds of no received messages.
poller.poll(max_number_of_messages:5, idle_timeout: 20) do |messages|
  messages.each do |msg|
    puts "Received message ID: #{msg.message_id}"
  end
end

# Check to see if Amazon SQS moved the message to the dead letter queue.
receive_message_result = sqs.receive_message({
  queue_url: dead_letter_queue_url,
  max_number_of_messages: 1
})

if receive_message_result.messages.count > 0
  puts "\n#{receive_message_result.messages[0].body}"
else
  puts "\nNo messages received."
end
```

Especificación del tiempo de espera de visibilidad de los mensajes en Amazon SQS

En Amazon SQS, inmediatamente después de recibirse un mensaje, este permanece en la cola. Para evitar que otros consumidores procesen el mensaje de nuevo, Amazon SQS establece un tiempo de espera de visibilidad. Se trata de un periodo de tiempo durante el cual Amazon SQS impide que otros componentes consumidores reciban y procesen el mensaje. Para obtener más información, consulte Tiempo de espera de visibilidad.

En este ejemplo, se utiliza AWS SDK para Ruby con Amazon SQS para:

1. Obtener la URL de una cola existente mediante Aws::SQS::Client#get_queue_url.
2. Recibir hasta 10 mensajes mediante Aws::SQS::Client#receive_message.
3. Especificar el intervalo de tiempo durante el cual los mensajes no son visibles después de recibirse mediante Aws::SQS::Client#change_message_visibility.

Requisitos previos

Antes de ejecutar el código de ejemplo, debe instalar y configurar AWS SDK para Ruby, tal y como se describe en:

- Instalación de AWS SDK para Ruby
- Configuración de AWS SDK para Ruby

También es necesario crear la cola my-queue, lo que puede hacer en la consola de Amazon SQS.

Ejemplo

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to specify the time interval during which messages to a queue are
 not visible after being received.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-east-1')

begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url
```

```
  receive_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
      queue_url: queue_url,
      receipt_handle: message.receipt_handle,
      visibility_timeout: 30 # This message will not be visible for 30 seconds after
 first receipt.
    })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
  })

  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."

rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{receive_queue_name}', as it does
 not exist."
end
```

Ejemplos de Amazon WorkDocs

Puede utilizar los siguientes ejemplos para acceder a Amazon WorkDocs (Amazon WorkDocs) mediante AWS SDK para Ruby. Para obtener más información sobre Amazon WorkDocs, consulte la documentación de Amazon WorkDocs.

Necesita su ID de organización para utilizar estos ejemplos. Obtenga el ID de su organización en la consola de AWSsiguiendo estos pasos:

- Seleccione AWS Directory Service

- Seleccionar `Directories`

El ID de organización es el `Directory ID` correspondiente a su sitio de Amazon WorkDocs.

Ejemplos

Temas

- Mostrar usuarios
- Lista de documentos del usuario

Mostrar usuarios

En el siguiente ejemplo se muestra una lista con los nombres, direcciones de correo electrónico y carpetas raíz de todos los usuarios de la organización. Elija `Copy` para guardar el código localmente o consulte el enlace al ejemplo completo al final de este tema.

1. Exija el módulo AWS SDK para Ruby y cree un cliente de Amazon WorkDocs.

2. Llame a `describe_users` con su ID de organización y obtenga todos los nombres de usuario en orden ascendente.

1. Muestre la información sobre los usuarios.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-workdocs'  # v2: require 'aws-sdk'

client = Aws::WorkDocs::Client.new(region: 'us-west-2')
```

```
# Set to the OrganizationId of your WorkDocs site
orgId = 'd-123456789c'

resp = client.describe_users({
  organization_id: orgId,
  include: "ALL", # accepts ALL, ACTIVE_PENDING
  order: "ASCENDING", # accepts ASCENDING, DESCENDING
  sort: "USER_NAME", # accepts USER_NAME, FULL_NAME, STORAGE_LIMIT, USER_STATUS,
 STORAGE_USED
})

resp.users.each do |user|
  puts "First name:  #{user.given_name}"
  puts "Last name:   #{user.surname}"
  puts "Email:       #{user.email_address}"
  puts "Root folder: #{user.root_folder_id}"
  puts
end
```

Consulte el ejemplo completo en GitHub.

Lista de documentos del usuario

En el ejemplo siguiente se muestran los documentos de un usuario. Elija Copy para guardar el código localmente o consulte el enlace al ejemplo completo al final de este tema.

1. Exija el módulo AWS SDK para Ruby.
2. Cree un método auxiliar para obtener la carpeta raíz de un usuario.
3. Cree un cliente de Amazon WorkDocs.
4. Obtenga la carpeta raíz para ese usuario.

5. Llame a `describe_folder_contents` para obtener el contenido de la carpeta en orden ascendente.

6. Muestra el nombre, el tamaño (en bytes), la fecha de la última modificación, el ID del documento y el ID de la versión de cada documento de la carpeta raíz del usuario.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
```

```
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-workdocs'  # v2: require 'aws-sdk'

def get_user_folder(client, orgId, user_email)
  root_folder = ''

  resp = client.describe_users({
    organization_id: orgId,
  })

  # resp.users should have only one entry
  resp.users.each do |user|
    if user.email_address == user_email
      root_folder = user.root_folder_id
    end
  end

  return root_folder
end

client = Aws::WorkDocs::Client.new(region: 'us-west-2')

# Set to the email address of a user
user_email = 'someone@somewhere'

# Set to the OrganizationId of your WorkDocs site.
orgId = 'd-123456789c'

user_folder = get_user_folder(client, orgId, user_email)

if user_folder == ''
  puts 'Could not get root folder for user with email address ' + user_email
  exit(1)
end

resp = client.describe_folder_contents({
  folder_id: user_folder, # required
```

```
  sort: "NAME", # accepts DATE, NAME
  order: "ASCENDING", # accepts ASCENDING, DESCENDING
})

resp.documents.each do |doc|
  md = doc.latest_version_metadata

  puts "Name:          #{md.name}"
  puts "Size (bytes):  #{md.size}"
  puts "Last modified: #{doc.modified_timestamp}"
  puts "Doc ID:        #{doc.id}"
  puts "Version ID:    #{md.id}"
  puts
end
```

Consulte el ejemplo completo en GitHub.

Ejemplos de código de SDK para Ruby

En los ejemplos de código de este tema se muestra cómo utilizar el AWS SDK for Ruby with AWS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Los ejemplos entre servicios son aplicaciones de muestra que funcionan en varios Servicios de AWS.

Ejemplos

- Acciones y escenarios con SDK para Ruby
- Ejemplos de servicios combinados con SDK para Ruby

Acciones y escenarios con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el comando AWS SDK for Ruby with Servicios de AWS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Servicios

- CloudTrail ejemplos de uso de SDK for Ruby
- CloudWatch ejemplos de uso de SDK for Ruby
- Ejemplos de DynamoDB con SDK para Ruby
- Ejemplos de Amazon EC2 con SDK para Ruby
- Ejemplos de Elastic Beanstalk con el SDK para Ruby

- EventBridge ejemplos de uso de SDK for Ruby
- AWS Glue ejemplos de uso de SDK for Ruby
- Ejemplos de IAM con SDK para Ruby
- Ejemplos de Kinesis con SDK for Ruby
- AWS KMS ejemplos de uso de SDK for Ruby
- Ejemplos de Lambda con SDK para Ruby
- Ejemplos de Amazon Polly con SDK for Ruby
- Ejemplos de Amazon RDS con SDK para Ruby
- Ejemplos de Amazon S3 con SDK para Ruby
- Ejemplos de Amazon SES que utilizan el SDK para Ruby
- Ejemplos de la API v2 de Amazon SES con SDK for Ruby
- Ejemplos de Amazon SNS con SDK para Ruby
- Ejemplos de Amazon SQS con SDK para Ruby
- AWS STS ejemplos de uso de SDK for Ruby
- WorkDocs Ejemplos de Amazon que utilizan SDK for Ruby

CloudTrail ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with CloudTrail.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Crea rutas

El siguiente ejemplo de código muestra cómo crear un AWS CloudTrail sendero.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-cloudtrail" # v2: require 'aws-sdk'
require "aws-sdk-s3"
require "aws-sdk-sts"

def create_trail_example(s3_client, sts_client, cloudtrail_client, trail_name,
 bucket_name)

  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to an Amazon Simple Storage Service (S3) bucket.
  s3_client.create_bucket(bucket: bucket_name)
  begin
    policy = {
      "Version" => "2012-10-17",
      "Statement" => [
        {
          "Sid" => "AWSCloudTrailAclCheck20150319",
          "Effect" => "Allow",
          "Principal" => {
            "Service" => "cloudtrail.amazonaws.com"
          },
          "Action" => "s3:GetBucketAcl",
          "Resource" => "arn:aws:s3:::#{bucket_name}"
        },
        {
          "Sid" => "AWSCloudTrailWrite20150319",
          "Effect" => "Allow",
```

```
          "Principal" => {
            "Service" => "cloudtrail.amazonaws.com"
          },
          "Action" => "s3:PutObject",
          "Resource" => "arn:aws:s3:::#{bucket_name}/AWSLogs/#{account_id}/*",
          "Condition" => {
            "StringEquals" => {
              "s3:x-amz-acl" => "bucket-owner-full-control"
            }
          }
        }
      ]
    }.to_json

    s3_client.put_bucket_policy(
      bucket: bucket_name,
      policy: policy
    )
    puts "Successfully added policy to bucket #{bucket_name}"
  end

  begin
    cloudtrail_client.create_trail({
                                     name: trail_name, # required
                                     s3_bucket_name: bucket_name # required
                                   })

    puts "Successfully created trail: #{trail_name}."
  rescue StandardError => e
    puts "Got error trying to create trail #{trail_name}:\n #{e}"
    puts e
    exit 1
  end
```

- Para obtener más información sobre la API, consulta CreateTraillla Referencia AWS SDK for Ruby de la API.

Eliminar rastro

El siguiente ejemplo de código muestra cómo eliminar una AWS CloudTrail ruta.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  client.delete_trail({
                        name: trail_name # required
                      })
  puts "Successfully deleted trail: " + trail_name
rescue StandardError => err
  puts "Got error trying to delete trail: " + trail_name + ":"
  puts err
  exit 1
end
```

- Para obtener más información sobre la API, consulta DeleteTraillla Referencia AWS SDK for Ruby de la API.

Enumere los eventos del sendero

En el siguiente ejemplo de código se muestra cómo enumerar los eventos de los AWS CloudTrail senderos.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-cloudtrail" # v2: require 'aws-sdk'

# @param [Object] client
def lookup_events_example(client)
  resp = client.lookup_events
```

```
  puts "Found #{resp.events.count} events:"
  resp.events.each do |e|
    puts "Event name:   #{e.event_name}"
    puts "Event ID:     #{e.event_id}"
    puts "Event time:   #{e.event_time}"
    puts "Resources:"

    e.resources.each do |r|
      puts "  Name:       #{r.resource_name}"
      puts "  Type:       #{r.resource_type}"
      puts ""
    end
  end
end
```

- Para obtener más información sobre la API, consulta LookupEventsla Referencia AWS SDK for Ruby de la API.

Enumere las rutas

El siguiente ejemplo de código muestra cómo enumerar las AWS CloudTrail rutas.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-cloudtrail"  # v2: require 'aws-sdk'

def describe_trails_example(client)
  resp = client.describe_trails({})
  puts "Found #{resp.trail_list.count} trail(s)."

  resp.trail_list.each do |trail|
    puts "Name:           " + trail.name
    puts "S3 bucket name: " + trail.s3_bucket_name
    puts
```

```
end
```

- Para obtener más información sobre la API, consulta ListTrailsla Referencia AWS SDK for Ruby de la API.

CloudWatch ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with CloudWatch.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Creación de una alarma para una métrica

El siguiente ejemplo de código muestra cómo crear o actualizar una CloudWatch alarma de Amazon y asociarla a la métrica especificada, la expresión matemática métrica, el modelo de detección de anomalías o la consulta de Metrics Insights especificados.

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
```

```
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  return true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  return false
end
```

- Para obtener más información sobre la API, consulta PutMetricAlarmla Referencia AWS SDK for Ruby de la API.

Descripción de alarmas

El siguiente ejemplo de código muestra cómo describir CloudWatch las alarmas de Amazon.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-cloudwatch"

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- Para obtener más información sobre la API, consulta DescribeAlarmsla Referencia AWS SDK for Ruby de la API.

Descripción de alarmas para una métrica

El siguiente ejemplo de código muestra cómo describir CloudWatch las alarmas de Amazon para una métrica.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts "-" * 16
      puts "Name:           " + alarm.alarm_name
      puts "State value:    " + alarm.state_value
      puts "State reason:   " + alarm.state_reason
      puts "Metric:         " + alarm.metric_name
      puts "Namespace:      " + alarm.namespace
      puts "Statistic:      " + alarm.statistic
      puts "Period:         " + alarm.period.to_s
      puts "Unit:           " + alarm.unit.to_s
      puts "Eval. periods:  " + alarm.evaluation_periods.to_s
      puts "Threshold:      " + alarm.threshold.to_s
      puts "Comp. operator: " + alarm.comparison_operator

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts "OK actions:"
        alarm.ok_actions.each do |a|
          puts "  " + a
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
        puts "Alarm actions:"
        alarm.alarm_actions.each do |a|
```

```
          puts "  " + a
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
          alarm.insufficient_data_actions.count.positive?
        puts "Insufficient data actions:"
        alarm.insufficient_data_actions.each do |a|
          puts "  " + a
        end
      end

      puts "Dimensions:"
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: " + d.name + ", Value: " + d.value
        end
      else
        puts "  None for this alarm."
      end
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ""

  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby cw-ruby-example-show-alarms.rb REGION"
    puts "Example: ruby cw-ruby-example-show-alarms.rb us-east-1"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = "us-east-1"
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
```

```
  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts "Available alarms:"
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta DescribeAlarmsForMetricla Referencia AWS SDK for Ruby de la API.

Deshabilitación de acciones de alarma

El siguiente ejemplo de código muestra cómo deshabilitar las acciones de CloudWatch alarma de Amazon.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
```

```
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  return true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  return false
end

# Example usage:
def run_me
  alarm_name = "ObjectsInBucket"
  alarm_description = "Objects exist in this bucket for more than 1 day."
  metric_name = "NumberOfObjects"
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ["arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic"]
  namespace = "AWS/S3"
  statistic = "Average"
  dimensions = [
    {
      name: "BucketName",
      value: "doc-example-bucket"
    },
    {
      name: "StorageType",
      value: "AllStorageTypes"
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = "Count"
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = "GreaterThanThreshold" # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
```

```
    alarm_actions,
    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

  if alarm_actions_disabled?(cloudwatch_client, alarm_name)
    puts "Alarm '#{alarm_name}' disabled."
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta DisableAlarmActionsla Referencia AWS SDK for Ruby de la API.

Enumerar métricas

El siguiente ejemplo de código muestra cómo enumerar los metadatos de CloudWatch las métricas de Amazon. Para obtener datos para una métrica, usa las GetMetricStatistics acciones GetMetricData o.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts "    Dimensions:"
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts "No dimensions found."
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      "Note that it could take up to 15 minutes for recently-added metrics " \
      "to become available."
  end
end

# Example usage:
def run_me
  metric_namespace = "SITE/TRAFFIC"
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
```

```
    metric_namespace,
    "UniqueVisitors",
    "SiteName",
    "example.com",
    5_885.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisits",
    "SiteName",
    "example.com",
    8_628.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "PageViews",
    "PageURL",
    "example.html",
    18_057.0,
    "Count"
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta ListMetricsla Referencia AWS SDK for Ruby de la API.

Colocar datos en una métrica

El siguiente ejemplo de código muestra cómo publicar puntos de datos métricos en Amazon CloudWatch.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-cloudwatch"

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
```

```
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  return true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  return false
end
```

- Para obtener más información sobre la API, consulta PutMetricDatala Referencia AWS SDK for Ruby de la API.

Ejemplos de DynamoDB con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante DynamoDB. AWS SDK for Ruby

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones
- Escenarios

Acciones

Creación de una tabla

En el siguiente ejemplo de código se muestra cómo crear una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource
  attr_reader :table_name
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
```

```
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        {attribute_name: "year", key_type: "HASH"},  # Partition key
        {attribute_name: "title", key_type: "RANGE"}  # Sort key
      ],
      attribute_definitions: [
        {attribute_name: "year", attribute_type: "N"},
        {attribute_name: "title", attribute_type: "S"}
      ],
      provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta CreateTablela Referencia AWS SDK for Ruby de la API.

Eliminación de una tabla

En el siguiente ejemplo de código se muestra cómo eliminar una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource
  attr_reader :table_name
  attr_reader :table
```

```
  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Deletes the table.
  def delete_table
    @table.delete
    @table = nil
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete table. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta DeleteTablela Referencia AWS SDK for Ruby de la API.

Eliminación de un elemento de una tabla

En el siguiente ejemplo de código se muestra cómo eliminar un elemento de una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
```

```
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
  # @param year [Integer] The release year of the movie to delete.
  def delete_item(title, year)
    @table.delete_item(key: {"year" => year, "title" => title})
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta DeleteItemla Referencia AWS SDK for Ruby de la API.

Obtención de un elemento de una tabla

En el siguiente ejemplo de código se muestra cómo obtener un elemento de una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
```

```
    @table = @dynamo_resource.table(table_name)
  end

  # Gets movie data from the table for a specific movie.
  #
  # @param title [String] The title of the movie.
  # @param year [Integer] The release year of the movie.
  # @return [Hash] The data about the requested movie.
  def get_item(title, year)
    @table.get_item(key: {"year" => year, "title" => title})
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't get movie #{title} (#{year}) from table #{@table.name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta GetItemla Referencia AWS SDK for Ruby de la API.

Obtener información sobre una tabla

En el siguiente ejemplo de código se muestra cómo obtener información sobre una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource
  attr_reader :table_name
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
```

```
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta DescribeTablela Referencia AWS SDK for Ruby de la API.

Mostrar tablas

En el siguiente ejemplo de código se muestra cómo enumerar las tablas de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Determinar si existe una tabla.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource
  attr_reader :table_name
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta ListTablesla Referencia AWS SDK for Ruby de la API.

Colocar un elemento en una tabla

En el siguiente ejemplo de código se muestra cómo colocar un elemento en una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Adds a movie to the table.
  #
  # @param movie [Hash] The title, year, plot, and rating of the movie.
  def add_item(movie)
    @table.put_item(
      item: {
        "year" => movie[:year],
        "title" => movie[:title],
        "info" => {"plot" => movie[:plot], "rating" => movie[:rating]}})
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta PutItemla Referencia AWS SDK for Ruby de la API.

Consultar una tabla

En el siguiente ejemplo de código se muestra cómo consultar una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.
  # @return [Array] The list of movies that were released in the specified year.
  def query_items(year)
    response = @table.query(
      key_condition_expression: "#yr = :year",
      expression_attribute_names: {"#yr" => "year"},
      expression_attribute_values: {":year" => year})
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't query for movies released in #{year}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.items
  end
```

- Para obtener información acerca de la API, consulte Query en la referencia de la API de AWS SDK for Ruby .

Ejecutar una instrucción PartiQL

En el siguiente ejemplo de código se muestra cómo ejecutar una instrucción PartiQL en una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Seleccionar un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Gets a single record from a table using PartiQL.
  # Note: To perform more fine-grained selects,
  # use the Client.query instance method instead.
  #
  # @param title [String] The title of the movie to search.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def select_item_by_title(title)
    request = {
      statement: "SELECT * FROM \"#{@table.name}\" WHERE title=?",
      parameters: [title]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Actualizar un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Updates a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def update_rating_by_title(title, year, rating)
    request = {
      statement: "UPDATE \"#{@table.name}\" SET info.rating=? WHERE title=? and
 year=?",
      parameters: [{ "N": rating }, title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Añadir un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Adds a single record to a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
```

```
  # @param year [Integer] The year the movie was released.
  # @param plot [String] The plot of the movie.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def insert_item(title, year, plot, rating)
    request = {
      statement: "INSERT INTO \"#{@table.name}\" VALUE {'title': ?, 'year': ?,
 'info': ?}",
      parameters: [title, year, {'plot': plot, 'rating': rating}]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Eliminar un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def delete_item_by_title(title, year)
    request = {
      statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
      parameters: [title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

- Para obtener más información sobre la API, consulta ExecuteStatementla Referencia AWS SDK for Ruby de la API.

Ejecutar lotes de instrucciones PartiQL

En el siguiente ejemplo de código se muestra cómo ejecutar lotes de instrucciones PartiQL en una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Leer un lote de elementos con PartiQL.

```
class DynamoDBPartiQLBatch

  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Selects a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_select(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "SELECT * FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({statements: request_items})
  end
```

Eliminar un lote de elementos con PartiQL.

```
class DynamoDBPartiQLBatch

  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_write(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({statements: request_items})
  end
```

- Para obtener más información sobre la API, consulta BatchExecuteStatementla Referencia AWS SDK for Ruby de la API.

Examinar una tabla

En el siguiente ejemplo de código, se muestra cómo examinar una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
    movies = []
    scan_hash = {
      filter_expression: "#yr between :start_yr and :end_yr",
      projection_expression: "#yr, title, info.rating",
      expression_attribute_names: {"#yr" => "year"},
      expression_attribute_values: {
        ":start_yr" => year_range[:start], ":end_yr" => year_range[:end]}
    }
    done = false
    start_key = nil
    until done
      scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
      response = @table.scan(scan_hash)
      movies.concat(response.items) unless response.items.empty?
      start_key = response.last_evaluated_key
      done = start_key.nil?
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't scan for movies. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    movies
  end
```

- Para obtener información acerca de la API, consulte Scan en la referencia de la API de AWS SDK for Ruby .

Actualizar un elemento en una tabla

En el siguiente ejemplo de código, se muestra cómo actualizar un elemento en una tabla de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Updates rating and plot data for a movie in the table.
  #
  # @param movie [Hash] The title, year, plot, rating of the movie.
  def update_item(movie)

    response = @table.update_item(
      key: {"year" => movie[:year], "title" => movie[:title]},
      update_expression: "set info.rating=:r",
      expression_attribute_values: { ":r" => movie[:rating] },
      return_values: "UPDATED_NEW")
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't update movie #{movie[:title]} (#{movie[:year]}) in table
 #{@table.name}\n")
    puts("\t#{e.code}: #{e.message}")
    raise
```

```
  else
    response.attributes
  end
```

- Para obtener más información sobre la API, consulta UpdateItemla Referencia AWS SDK for Ruby de la API.

Escribir un lote de elementos

En el siguiente ejemplo de código, se muestra cómo escribir un lote de elementos de DynamoDB.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: "us-east-1")
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Fills an Amazon DynamoDB table with the specified data. Items are sent in
  # batches of 25 until all items are written.
  #
  # @param movies [Enumerable] The data to put in the table. Each item must contain
 at least
  #                            the keys required by the schema that was specified
 when the
  #                            table was created.
  def write_batch(movies)
    index = 0
    slice_size = 25
    while index < movies.length
```

```
      movie_items = []
      movies[index, slice_size].each do |movie|
        movie_items.append({put_request: { item: movie }})
      end
      @dynamo_resource.client.batch_write_item({request_items: { @table.name =>
movie_items }})
      index += slice_size
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts(
      "Couldn't load data into table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta BatchWriteItemla Referencia AWS SDK for Ruby de la API.

Escenarios

Introducción a tablas, elementos y consultas

En el siguiente ejemplo de código, se muestra cómo:

- Creación de una tabla que pueda contener datos de películas.
- Colocar, obtener y actualizar una sola película en la tabla.
- Escribir los datos de películas en la tabla a partir de un archivo JSON de ejemplo.
- Consultar películas que se hayan estrenado en un año determinado.
- Buscar películas que se hayan estrenado en un intervalo de años.
- Eliminación de una película de la tabla y, a continuación, eliminar la tabla.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crear una clase que encapsula una tabla de DynamoDB.

```
  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        {attribute_name: "year", key_type: "HASH"},  # Partition key
        {attribute_name: "title", key_type: "RANGE"}  # Sort key
      ],
      attribute_definitions: [
        {attribute_name: "year", attribute_type: "N"},
        {attribute_name: "title", attribute_type: "S"}
      ],
      provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

Crear una función auxiliar para descargar y extraer el archivo JSON de muestra.

```
  # Gets sample movie data, either from a local file or by first downloading it from
  # the Amazon DynamoDB Developer Guide.
  #
  # @param movie_file_name [String] The local file name where the movie data is
 stored in JSON format.
  # @return [Hash] The movie data as a Hash.
  def fetch_movie_data(movie_file_name)
    if !File.file?(movie_file_name)
      @logger.debug("Downloading #{movie_file_name}...")
      movie_content = URI.open(
        "https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
moviedata.zip"
      )
```

```
      movie_json = ""
      Zip::File.open_buffer(movie_content) do |zip|
        zip.each do |entry|
          movie_json = entry.get_input_stream.read
        end
      end
    else
      movie_json = File.read(movie_file_name)
    end
    movie_data = JSON.parse(movie_json)
    # The sample file lists over 4000 movies. This returns only the first 250.
    movie_data.slice(0, 250)
  rescue StandardError => e
    puts("Failure downloading movie data:\n#{e}")
    raise
  end
```

Ejecutar un escenario interactivo para crear la tabla y realizar acciones en ella.

```
  table_name = "doc-example-table-movies-#{rand(10**4)}"
  scaffold = Scaffold.new(table_name)
  dynamodb_wrapper = DynamoDBBasics.new(table_name)

  new_step(1, "Create a new DynamoDB table if none already exists.")
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, "Add a new record to the DynamoDB table.")
  my_movie = {}
  my_movie[:title] = CLI::UI::Prompt.ask("Enter the title of a movie to add to the
table. E.g. The Matrix")
  my_movie[:year] = CLI::UI::Prompt.ask("What year was it released? E.g. 1989").to_i
  my_movie[:rating] = CLI::UI::Prompt.ask("On a scale of 1 - 10, how do you rate it?
E.g. 7").to_i
  my_movie[:plot] = CLI::UI::Prompt.ask("Enter a brief summary of the plot. E.g. A
man awakens to a new reality.")
  dynamodb_wrapper.add_item(my_movie)
  puts("\nNew record added:")
  puts JSON.pretty_generate(my_movie).green
```

```
 print "Done!\n".green

 new_step(3, "Update a record in the DynamoDB table.")
 my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with a
new rating, e.g. 3:").to_i
 response = dynamodb_wrapper.update_item(my_movie)
 puts("Updated '#{my_movie[:title]}' with new attributes:")
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(4, "Get a record from the DynamoDB table.")
 puts("Searching for #{my_movie[:title]} (#{my_movie[:year]})...")
 response = dynamodb_wrapper.get_item(my_movie[:title], my_movie[:year])
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(5, "Write a batch of items into the DynamoDB table.")
 download_file = "moviedata.json"
 puts("Downloading movie database to #{download_file}...")
 movie_data = scaffold.fetch_movie_data(download_file)
 puts("Writing movie data from #{download_file} into your table...")
 scaffold.write_batch(movie_data)
 puts("Records added: #{movie_data.length}.")
 print "Done!\n".green

 new_step(5, "Query for a batch of items by key.")
 loop do
   release_year = CLI::UI::Prompt.ask("Enter a year between 1972 and 2018, e.g.
1999:").to_i
   results = dynamodb_wrapper.query_items(release_year)
   if results.any?
     puts("There were #{results.length} movies released in #{release_year}:")
     results.each do |movie|
       print "\t #{movie["title"]}".green
     end
     break
   else
     continue = CLI::UI::Prompt.ask("Found no movies released in #{release_year}!
Try another year? (y/n)")
     break if !continue.eql?("y")
   end
 end
 print "\nDone!\n".green
```

```
  new_step(6, "Scan for a batch of items using a filter expression.")
  years = {}
  years[:start] = CLI::UI::Prompt.ask("Enter a starting year between 1972 and
 2018:")
  years[:end] = CLI::UI::Prompt.ask("Enter an ending year between 1972 and 2018:")
  releases = dynamodb_wrapper.scan_items(years)
  if !releases.empty?
    puts("Found #{releases.length} movies.")
    count = Question.ask(
      "How many do you want to see? ", method(:is_int), in_range(1,
 releases.length))
    puts("Here are your #{count} movies:")
    releases.take(count).each do |release|
      puts("\t#{release["title"]}")
    end
  else
    puts("I don't know about any movies released between #{years[:start]} "\
         "and #{years[:end]}.")
  end
  print "\nDone!\n".green

  new_step(7, "Delete an item from the DynamoDB table.")
  answer = CLI::UI::Prompt.ask("Do you want to remove '#{my_movie[:title]}'? (y/n)
")
  if answer.eql?("y")
    dynamodb_wrapper.delete_item(my_movie[:title], my_movie[:year])
    puts("Removed '#{my_movie[:title]}' from the table.")
    print "\nDone!\n".green
  end

  new_step(8, "Delete the DynamoDB table.")
  answer = CLI::UI::Prompt.ask("Delete the table? (y/n)")
  if answer.eql?("y")
    scaffold.delete_table
    puts("Deleted #{table_name}.")
  else
    puts("Don't forget to delete the table when you're done!")
  end
  print "\nThanks for watching!\n".green
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo.")
rescue Errno::ENOENT
  true
end
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Ruby .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Consultar una tabla mediante lotes de instrucciones PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtención de un lote de elementos mediante la ejecución de varias instrucciones SELECT.
- Agregar un lote de elementos mediante la ejecución de varias instrucciones INSERT.
- Actualizar un lote de elementos con la ejecución de varias instrucciones UPDATE.
- Eliminación de un lote de elementos con la ejecución de varias instrucciones DELETE.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario que crea una tabla y ejecuta lotes de consultas PartiQL.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**4)}"
```

```
  scaffold = Scaffold.new(table_name)
  sdk = DynamoDBPartiQLBatch.new(table_name)

  new_step(1, "Create a new DynamoDB table if none already exists.")
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, "Populate DynamoDB table with movie data.")
  download_file = "moviedata.json"
  puts("Downloading movie database to #{download_file}...")
  movie_data = scaffold.fetch_movie_data(download_file)
  puts("Writing movie data from #{download_file} into your table...")
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(3, "Select a batch of items from the movies table.")
  puts "Let's select some popular movies for side-by-side comparison."
  response = sdk.batch_execute_select([["Mean Girls", 2004], ["Goodfellas", 1977],
 ["The Prancing of the Lambs", 2005]])
  puts("Items selected: #{response['responses'].length}\n")
  print "\nDone!\n".green

  new_step(4, "Delete a batch of items from the movies table.")
  sdk.batch_execute_write([["Mean Girls", 2004], ["Goodfellas", 1977], ["The
 Prancing of the Lambs", 2005]])
  print "\nDone!\n".green

  new_step(5, "Delete the table.")
  if scaffold.exists?(table_name)
    scaffold.delete_table
  end
end
```

- Para obtener más información sobre la API, consulta BatchExecuteStatementla Referencia AWS SDK for Ruby de la API.

Consultar una tabla con PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtención de un artículo mediante una instrucción SELECT.
- Agregar un elemento mediante una instrucción INSERT.
- Actualizar un elemento mediante una instrucción UPDATE.
- Eliminación de un elemento mediante una instrucción DELETE.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario que crea una tabla y ejecuta consultas PartiQL.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**8)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLSingle.new(table_name)

new_step(1, "Create a new DynamoDB table if none already exists.")
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end

new_step(2, "Populate DynamoDB table with movie data.")
download_file = "moviedata.json"
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
scaffold.write_batch(movie_data)
puts("Records added: #{movie_data.length}.")
print "Done!\n".green

new_step(3, "Select a single item from the movies table.")
response = sdk.select_item_by_title("Star Wars")
```

```
  puts("Items selected for title 'Star Wars': #{response.items.length}\n")
  print "#{response.items.first}".yellow
  print "\n\nDone!\n".green

  new_step(4, "Update a single item from the movies table.")
  puts "Let's correct the rating on The Big Lebowski to 10.0."
  sdk.update_rating_by_title("The Big Lebowski", 1998, 10.0)
  print "\nDone!\n".green

  new_step(5, "Delete a single item from the movies table.")
  puts "Let's delete The Silence of the Lambs because it's just too scary."
  sdk.delete_item_by_title("The Silence of the Lambs", 1991)
  print "\nDone!\n".green

  new_step(6, "Insert a new item into the movies table.")
  puts "Let's create a less-scary movie called The Prancing of the Lambs."
  sdk.insert_item("The Prancing of the Lambs", 2005, "A movie about happy
 livestock.", 5.0)
  print "\nDone!\n".green

  new_step(7, "Delete the table.")
  if scaffold.exists?(table_name)
    scaffold.delete_table
  end
end
```

- Para obtener más información sobre la API, consulta ExecuteStatementla Referencia AWS SDK for Ruby de la API.

Ejemplos de Amazon EC2 con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Amazon EC2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Asignar una dirección IP elástica

En el siguiente ejemplo de código se muestra cómo asignar una dirección IP elástica de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: "vpc")
  return response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  return "Error"
end
```

- Para obtener más información sobre la API, consulta AllocateAddressla Referencia AWS SDK for Ruby de la API.

Asociación de una dirección IP elástica a una instancia

En el siguiente ejemplo de código se muestra cómo asociar una dirección IP elástica a una instancia de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id,
  )
  return response.association_id
```

```
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  return "Error"
end
```

- Para obtener más información sobre la API, consulta AssociateAddressla Referencia AWS SDK for Ruby de la API.

Crear una Amazon Virtual Private Cloud (Amazon VPC)

En el siguiente ejemplo de código, se muestra cómo crear una Amazon Virtual Private Cloud (Amazon VPC).

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ec2"

# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
```

```
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "#{e.message}"
  return false
end

# Example usage:
def run_me
  cidr_block = ""
  tag_key = ""
  tag_value = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-create-vpc.rb " \
      "CIDR_BLOCK TAG_KEY TAG_VALUE REGION"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-create-vpc.rb " \
      "10.0.0.0/24 my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = "10.0.0.0/24"
    tag_key = "my-key"
    tag_value = "my-value"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
```

```
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,
    cidr_block,
    tag_key,
    tag_value
  )
    puts "VPC created and tagged."
  else
    puts "VPC not created or not tagged."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta CreateVpcla Referencia AWS SDK for Ruby de la API.

Crear una tabla de enrutamiento

En el siguiente ejemplo de código se muestra cómo crear una tabla de enrutamiento y asociarla a la subred de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ec2"

# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
```

```
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Added tags to route table."
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts "Created route with destination CIDR block " \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  return true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts "If the route table was created but not associated, you should " \
    "clean up by deleting the route table."
  return false
end

# Example usage:
def run_me
  vpc_id = ""
  subnet_id = ""
  gateway_id = ""
  destination_cidr_block = ""
  tag_key = ""
  tag_value = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage: ruby ec2-ruby-example-create-route-table.rb " \
      "VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK " \
      "TAG_KEY TAG_VALUE REGION"
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-create-route-table.rb " \
      "vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE " \
      "'0.0.0.0/0' my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
```

```
    vpc_id = "vpc-0b6f769731EXAMPLE"
    subnet_id = "subnet-03d9303b57EXAMPLE"
    gateway_id = "igw-06ca90c011EXAMPLE"
    destination_cidr_block = "0.0.0.0/0"
    tag_key = "my-key"
    tag_value = "my-value"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    subnet_id = ARGV[1]
    gateway_id = ARGV[2]
    destination_cidr_block = ARGV[3]
    tag_key = ARGV[4]
    tag_value = ARGV[5]
    region = ARGV[6]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if route_table_created_and_associated?(
    ec2_resource,
    vpc_id,
    subnet_id,
    gateway_id,
    destination_cidr_block,
    tag_key,
    tag_value
  )
    puts "Route table created and associated."
  else
    puts "Route table not created or not associated."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta CreateRouteTablela Referencia AWS SDK for Ruby de la API.

Creación de un grupo de seguridad

En el siguiente ejemplo de código se muestra cómo crear un grupo de seguridad de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require "aws-sdk-ec2"

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(
  ec2_client,
  group_name,
```

```
  description,
  vpc_id
)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  return security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  return "Error"
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client,
  security_group_id,
  ip_protocol,
  from_port,
```

```
  to_port,
  cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  return true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  return false
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - A security group with inbound rules, outbound rules, or both.
#
# @param p [Aws::EC2::Types::IpPermission] The IP permissions set.
# @example
#   ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
#   response = ec2_client.describe_security_groups
#   unless sg.ip_permissions.empty?
#     describe_security_group_permissions(
#       response.security_groups[0].ip_permissions[0]
#     )
#   end
def describe_security_group_permissions(perm)
```

```
  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"

  unless perm.from_port.nil?
    if perm.from_port == "-1" || perm.from_port == -1
      print ", From: All"
    else
      print ", From: #{perm.from_port}"
    end
  end

  unless perm.to_port.nil?
    if perm.to_port == "-1" || perm.to_port == -1
      print ", To: All"
    else
      print ", To: #{perm.to_port}"
    end
  end

  if perm.key?(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?
    print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}"
  end

  if perm.key?(:ip_ranges) && perm.ip_ranges.count.positive?
    print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}"
  end

  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @example
#   describe_security_groups(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      puts "-" * (sg.group_name.length + 13)
      puts "Name:        #{sg.group_name}"
      puts "Description: #{sg.description}"
      puts "Group ID:    #{sg.group_id}"
```

```
      puts "Owner ID:    #{sg.owner_id}"
      puts "VPC ID:      #{sg.vpc_id}"

      if sg.tags.count.positive?
        puts "Tags:"
        sg.tags.each do |tag|
          puts "  Key: #{tag.key}, Value: #{tag.value}"
        end
      end

      unless sg.ip_permissions.empty?
        puts "Inbound rules:" if sg.ip_permissions.count.positive?
        sg.ip_permissions.each do |p|
          describe_security_group_permissions(p)
        end
      end

      unless sg.ip_permissions_egress.empty?
        puts "Outbound rules:" if sg.ip_permissions.count.positive?
        sg.ip_permissions_egress.each do |p|
          describe_security_group_permissions(p)
        end
      end
    end
  else
    puts "No security groups found."
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param security_group_id [String] The ID of the security group to delete.
# @return [Boolean] true if the security group was deleted; otherwise, false.
# @example
#   exit 1 unless security_group_deleted?(
```

```
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX'
#   )
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  return true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  return false
end

# Example usage:
def run_me
  group_name = ""
  description = ""
  vpc_id = ""
  ip_protocol_http = ""
  from_port_http = ""
  to_port_http = ""
  cidr_ip_range_http = ""
  ip_protocol_ssh = ""
  from_port_ssh = ""
  to_port_ssh = ""
  cidr_ip_range_ssh = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-security-group.rb " \
      "GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 " \
      "CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 " \
      "CIDR_IP_RANGE_2 REGION"
    puts "Example: ruby ec2-ruby-example-security-group.rb " \
      "my-security-group 'This is my security group.' vpc-6713dfEX " \
      "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    group_name = "my-security-group"
    description = "This is my security group."
    vpc_id = "vpc-6713dfEX"
    ip_protocol_http = "tcp"
    from_port_http = "80"
    to_port_http = "80"
```

```
  cidr_ip_range_http = "0.0.0.0/0"
  ip_protocol_ssh = "tcp"
  from_port_ssh = "22"
  to_port_ssh = "22"
  cidr_ip_range_ssh = "0.0.0.0/0"
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  group_name = ARGV[0]
  description = ARGV[1]
  vpc_id = ARGV[2]
  ip_protocol_http = ARGV[3]
  from_port_http = ARGV[4]
  to_port_http = ARGV[5]
  cidr_ip_range_http = ARGV[6]
  ip_protocol_ssh = ARGV[7]
  from_port_ssh = ARGV[8]
  to_port_ssh = ARGV[9]
  cidr_ip_range_ssh = ARGV[10]
  region = ARGV[11]
end

security_group_id = ""
security_group_exists = false
ec2_client = Aws::EC2::Client.new(region: region)

puts "Attempting to create security group..."
security_group_id = create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
if security_group_id == "Error"
  puts "Could not create security group. Skipping this step."
else
  security_group_exists = true
end

if security_group_exists
  puts "Attempting to add inbound rules to security group..."
  unless security_group_ingress_authorized?(
    ec2_client,
```

```
      security_group_id,
      ip_protocol_http,
      from_port_http,
      to_port_http,
      cidr_ip_range_http
    )
      puts "Could not add inbound HTTP rule to security group. " \
        "Skipping this step."
    end

    unless security_group_ingress_authorized?(
      ec2_client,
      security_group_id,
      ip_protocol_ssh,
      from_port_ssh,
      to_port_ssh,
      cidr_ip_range_ssh
    )
      puts "Could not add inbound SSH rule to security group. " \
        "Skipping this step."
    end
  end

  puts "\nInformation about available security groups:"
  describe_security_groups(ec2_client)

  if security_group_exists
    puts "\nAttempting to delete security group..."
    unless security_group_deleted?(ec2_client, security_group_id)
      puts "Could not delete security group. You must delete it yourself."
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta CreateSecurityGroupla Referencia AWS SDK for Ruby de la API.

Creación de un par de claves de seguridad

En el siguiente ejemplo de código se muestra cómo crear un par de claves de seguridad de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require "aws-sdk-ec2"

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, key_pair_name + ".pem")
  File.open(filename, "w") { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  return true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    "already exists."
```

```
  return false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  return false
end

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts "No key pairs found."
  else
    puts "Key pair names:"
    result.key_pairs.each do |key_pair|
      puts key_pair.key_name
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  return true
rescue StandardError => e
```

```
  puts "Error deleting key pair: #{e.message}"
  return false
end

# Example usage:
def run_me
  key_pair_name = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION"
    puts "Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = "my-key-pair"
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Displaying existing key pair names before creating this key pair..."
  describe_key_pairs(ec2_client)

  puts "-" * 10
  puts "Creating key pair..."
  unless key_pair_created?(ec2_client, key_pair_name)
    puts "Stopping program."
    exit 1
  end

  puts "-" * 10
  puts "Displaying existing key pair names after creating this key pair..."
  describe_key_pairs(ec2_client)

  puts "-" * 10
  puts "Deleting key pair..."
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts "Stopping program. You must delete the key pair yourself."
```

```
    exit 1
  end
  puts "Key pair deleted."

  puts "-" * 10
  puts "Now that the key pair is deleted, " \
    "also deleting the related private key pair file..."
  filename = File.join(Dir.home, key_pair_name + ".pem")
  File.delete(filename)
  if File.exist?(filename)
    puts "Could not delete file at '#{filename}'. You must delete it yourself."
  else
    puts "File deleted."
  end

  puts "-" * 10
  puts "Displaying existing key pair names after deleting this key pair..."
  describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta CreateKeyPairla Referencia AWS SDK for Ruby de la API.

Crear una subred

En los siguientes ejemplos de código, se muestra cómo crear una subred de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ec2"

# Creates a subnet within a virtual private cloud (VPC) in
```

```
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
```

```
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  return false
end

# Example usage:
def run_me
  vpc_id = ""
  cidr_block = ""
  availability_zone = ""
  tag_key = ""
  tag_value = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-create-subnet.rb " \
      "VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-create-subnet.rb " \
      "vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = "vpc-6713dfEX"
    cidr_block = "10.0.0.0/24"
    availability_zone = "us-west-2a"
    tag_key = "my-key"
    tag_value = "my-value"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
    tag_key = ARGV[3]
```

```
    tag_value = ARGV[4]
    region = ARGV[5]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if subnet_created_and_tagged?(
    ec2_resource,
    vpc_id,
    cidr_block,
    availability_zone,
    tag_key,
    tag_value
  )
    puts "Subnet created and tagged."
  else
    puts "Subnet not created or not tagged."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta CreateSubnetla Referencia AWS SDK for Ruby de la API.

Describir regiones

En el siguiente ejemplo de código, se muestra cómo describir regiones de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ec2"

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
```

```
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print "Region"
  print " " * (max_region_string_length - "Region".length)
  print "  Endpoint\n"
  print "-" * max_region_string_length
  print "  "
  print "-" * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print " " * (max_region_string_length - region.region_name.length)
    print "  "
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print "Region"
  print " " * (max_region_string_length - "Region".length)
  print "  Zone"
  print " " * (max_zone_string_length - "Zone".length)
  print "  State\n"
```

```
  print "-" * max_region_string_length
  print "  "
  print "-" * max_zone_string_length
  print "  "
  print "-" * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print " " * (max_region_string_length - zone.region_name.length)
    print "  "
    print zone.zone_name
    print " " * (max_zone_string_length - zone.zone_name.length)
    print "  "
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts "  Messages for this zone:"
      zone.messages.each do |message|
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end

# Example usage:
def run_me
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-regions-availability-zones.rb REGION"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-regions-availability-zones.rb us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
```

```
  ec2_client = Aws::EC2::Client.new(region: region)

  puts "AWS Regions for Amazon EC2 that are available to you:"
  list_regions_endpoints(ec2_client)
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS Region
 '#{region}':"
  list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta DescribeRegionsla Referencia AWS SDK for Ruby de la API.

Descripción de instancias

En el siguiente ejemplo de código se muestra cómo describir instancias de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ec2"

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts "No instances found."
  else
    puts "Instances -- ID, state:"
    response.each do |instance|
```

```
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta DescribeInstancesla Referencia AWS SDK for Ruby de la API.

Liberar una dirección IP elástica

En el siguiente ejemplo de código se muestra cómo liberar una dirección IP elástica.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  return true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  return false
end
```

- Para obtener más información sobre la API, consulta ReleaseAddressla Referencia AWS SDK for Ruby de la API.

Inicio de una instancia

En el siguiente ejemplo de código se muestra cómo iniciar una instancia de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ec2"

# Attempts to start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was started; otherwise, false.
# @example
#   exit 1 unless instance_started?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when "pending"
      puts "Error starting instance: the instance is pending. Try again later."
      return false
    when "running"
      puts "The instance is already running."
      return true
    when "terminated"
      puts "Error starting instance: " \
        "the instance is terminated, so you cannot start it."
      return false
    end
```

```
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts "Instance started."
  return true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  return false
end

# Example usage:
def run_me
  instance_id = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-start-instance-i-123abc.rb " \
      "INSTANCE_ID REGION "
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-start-instance-i-123abc.rb " \
      "i-123abc us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = "i-123abc"
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to start instance '#{instance_id}' " \
    "(this might take a few minutes)..."
  unless instance_started?(ec2_client, instance_id)
    puts "Could not start instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta StartInstancesla Referencia AWS SDK for Ruby de la API.

Detener una instancia

En el siguiente ejemplo de código se muestra cómo detener una instancia de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ec2"

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when "stopping"
      puts "The instance is already stopping."
      return true
    when "stopped"
```

```
      puts "The instance is already stopped."
      return true
    when "terminated"
      puts "Error stopping instance: " \
        "the instance is terminated, so you cannot stop it."
      return false
    end
  end

  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts "Instance stopped."
  return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return false
end

# Example usage:
def run_me
  instance_id = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-stop-instance-i-123abc.rb " \
      "INSTANCE_ID REGION "
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-start-instance-i-123abc.rb " \
      "i-123abc us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = "i-123abc"
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
```

```
    "(this might take a few minutes)..."
  unless instance_stopped?(ec2_client, instance_id)
    puts "Could not stop instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta StopInstancesla Referencia AWS SDK for Ruby de la API.

Finalizar una instancia

En el siguiente ejemplo de código se muestra cómo terminar una instancia de Amazon EC2.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ec2"

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])
```

```
  if response.instance_statuses.count.positive? &&
    response.instance_statuses[0].instance_state.name == "terminated"

    puts "The instance is already terminated."
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts "Instance terminated."
  return true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  return false
end

# Example usage:
def run_me
  instance_id = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-terminate-instance-i-123abc.rb " \
      "INSTANCE_ID REGION "
   # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb " \
      "i-123abc us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = "i-123abc"
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to terminate instance '#{instance_id}' " \
    "(this might take a few minutes)..."
  unless instance_terminated?(ec2_client, instance_id)
```

```
    puts "Could not terminate instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta TerminateInstancesla Referencia AWS SDK for Ruby de la API.

Ejemplos de Elastic Beanstalk con el SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby con Elastic Beanstalk.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Describe la aplicación

El siguiente ejemplo de código muestra cómo describir una AWS Elastic Beanstalk aplicación.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Class to manage Elastic Beanstalk applications
class ElasticBeanstalkManager
  def initialize(eb_client, logger: Logger.new($stdout))
    @eb_client = eb_client
    @logger = logger
  end

  # Lists applications and their environments
  def list_applications
    @eb_client.describe_applications.applications.each do |application|
      log_application_details(application)
      list_environments(application.application_name)
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Elastic Beanstalk Service Error: #{e.message}")
  end

  private

  # Logs application details
  def log_application_details(application)
    @logger.info("Name:        #{application.application_name}")
    @logger.info("Description: #{application.description}")
  end

  # Lists and logs details of environments for a given application
  def list_environments(application_name)
    @eb_client.describe_environments(application_name:
 application_name).environments.each do |env|
      @logger.info("  Environment:  #{env.environment_name}")
      @logger.info("    URL:        #{env.cname}")
      @logger.info("    Health:     #{env.health}")
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
```

```
    @logger.error("Error listing environments for application #{application_name}:
 #{e.message}")
  end
end
```

- Para obtener más información sobre la API, consulta DescribeApplicationsla Referencia AWS SDK for Ruby de la API.

Enumere las pilas

El siguiente ejemplo de código muestra cómo enumerar las AWS Elastic Beanstalk pilas.

SDK para Ruby

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Manages listing of AWS Elastic Beanstalk solution stacks
# @param [Aws::ElasticBeanstalk::Client] eb_client
# @param [String] filter - Returns subset of results based on match
# @param [Logger] logger
class StackLister
  # Initialize with AWS Elastic Beanstalk client
  def initialize(eb_client, filter, logger: Logger.new($stdout))
    @eb_client = eb_client
    @filter = filter.downcase
    @logger = logger
  end

  # Lists and logs Elastic Beanstalk solution stacks
  def list_stacks
    stacks = @eb_client.list_available_solution_stacks.solution_stacks
    orig_length = stacks.length
    filtered_length = 0

    stacks.each do |stack|
      if @filter.empty? || stack.downcase.include?(@filter)
        @logger.info(stack)
```

```
        filtered_length += 1
      end
    end

    log_summary(filtered_length, orig_length)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Error listing solution stacks: #{e.message}")
  end

  private

  # Logs summary of listed stacks
  def log_summary(filtered_length, orig_length)
    if @filter.empty?
      @logger.info("Showed #{orig_length} stack(s)")
    else
      @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
    end
  end
end
```

- Para obtener más información sobre la API, consulta ListAvailableSolutionStacksla Referencia AWS SDK for Ruby de la API.

Actualiza la aplicación

El siguiente ejemplo de código muestra cómo actualizar una AWS Elastic Beanstalk aplicación.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Manages deployment of Rails applications to AWS Elastic Beanstalk
class RailsAppDeployer
  def initialize(eb_client, s3_client, app_name, logger: Logger.new($stdout))
    @eb_client = eb_client
    @s3_client = s3_client
```

```
    @app_name = app_name
    @logger = logger
  end

  # Deploys the latest application version to Elastic Beanstalk
  def deploy
    create_storage_location
    zip_file_name = create_zip_file
    upload_zip_to_s3(zip_file_name)
    create_and_deploy_new_application_version(zip_file_name)
  end

  private

  # Creates a new S3 storage location for the application
  def create_storage_location
    resp = @eb_client.create_storage_location
    @logger.info("Created storage location in bucket #{resp.s3_bucket}")
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create storage location: #{e.message}")
  end

  # Creates a ZIP file of the application using git
  def create_zip_file
    zip_file_basename = SecureRandom.urlsafe_base64
    zip_file_name = "#{zip_file_basename}.zip"
    `git archive --format=zip -o #{zip_file_name} HEAD`
    zip_file_name
  end

  # Uploads the ZIP file to the S3 bucket
  def upload_zip_to_s3(zip_file_name)
    zip_contents = File.read(zip_file_name)
    key = "#{@app_name}/#{zip_file_name}"
    @s3_client.put_object(body: zip_contents, bucket: fetch_bucket_name, key: key)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to upload ZIP file to S3: #{e.message}")
  end

  # Fetches the S3 bucket name from Elastic Beanstalk application versions
  def fetch_bucket_name
    app_versions = @eb_client.describe_application_versions(application_name:
@app_name)
    av = app_versions.application_versions.first
```

```
    av.source_bundle.s3_bucket
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch bucket name: #{e.message}")
    raise
  end

  # Creates a new application version and deploys it
  def create_and_deploy_new_application_version(zip_file_name)
    version_label = File.basename(zip_file_name, ".zip")
    @eb_client.create_application_version(
      process: false,
      application_name: @app_name,
      version_label: version_label,
      source_bundle: {
        s3_bucket: fetch_bucket_name,
        s3_key: "#{@app_name}/#{zip_file_name}"
      },
      description: "Updated #{Time.now.strftime('%d/%m/%Y')}"
    )
    update_environment(version_label)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create or deploy application version: #{e.message}")
  end

  # Updates the environment to the new application version
  def update_environment(version_label)
    env_name = fetch_environment_name
    @eb_client.update_environment(
      environment_name: env_name,
      version_label: version_label
    )
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to update environment: #{e.message}")
  end

  # Fetches the environment name of the application
  def fetch_environment_name
    envs = @eb_client.describe_environments(application_name: @app_name)
    envs.environments.first.environment_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch environment name: #{e.message}")
    raise
  end
end
```

- Para obtener más información sobre la API, consulta UpdateApplicationla Referencia AWS SDK for Ruby de la API.

EventBridge ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with EventBridge.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Escenarios

Escenarios

Crear y activar una regla

El siguiente ejemplo de código muestra cómo crear y activar una regla en Amazon EventBridge.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Llamar a las funciones en el orden correcto.

```
require "aws-sdk-sns"
require "aws-sdk-iam"
require "aws-sdk-cloudwatchevents"
require "aws-sdk-ec2"
require "aws-sdk-cloudwatch"
require "aws-sdk-cloudwatchlogs"
require "securerandom"
```

Comprobar si el tema de Amazon Simple Notification Service (Amazon SNS) especificado existe entre los que se proporcionan para esta función.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end

def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  return false
end
```

Comprobar si el tema especificado existe entre los disponibles para el intermediario en Amazon SNS.

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
```

```
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts "Topic found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts "Topic found."
          return true
        end
      end
    end
  end
  puts "Topic not found."
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
end
```

Crear un tema en Amazon SNS y después suscribe una dirección de correo electrónico para recibir notificaciones sobre dicho tema.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
```

```
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: "email",
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts "Subscription created with ARN " \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    "and confirm the subscription to start receiving notification emails."
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return "Error"
end
```

Compruebe si el rol especificado AWS Identity and Access Management (IAM) existe entre los que se proporcionan a esta función.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
```

```
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end
```

Comprobar si el rol especificado existe entre los disponibles para el intermediario en IAM.

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts "Role found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.roles.count.positive?
        if role_found?(response.roles, role_arn)
          puts "Role found."
          return true
        end
      end
    end
```

```
  end
  puts "Role not found."
  return false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  return false
end
```

Crear un rol en IAM.

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "",
          'Effect': "Allow",
          'Principal': {
            'Service': "events.amazonaws.com"
          },
          'Action': "sts:AssumeRole"
        }
      ]
    }.to_json,
    path: "/",
    role_name: role_name
  )
```

```
  puts "Role created with ARN '#{response.role.arn}'."
  puts "Adding access policy to role..."
  iam_client.put_role_policy(
    policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "CloudWatchEventsFullAccess",
          'Effect': "Allow",
          'Resource': "*",
          'Action': "events:*"
        },
        {
          'Sid': "IAMPassRoleForCloudWatchEvents",
          'Effect': "Allow",
          'Resource': "arn:aws:iam::*:role/AWS_Events_Invoke_Targets",
          'Action': "iam:PassRole"
        }
      ]
    }.to_json,
    policy_name: "CloudWatchEventsPolicy",
    role_name: role_name
  )
  puts "Access policy added to role."
  return response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts "If the role was created, you must add the access policy " \
    "to the role yourself, or delete the role yourself and try again."
  return "Error"
end
```

Comprueba si la EventBridge regla especificada existe entre las que se proporcionan a esta función.

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
```

```
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  return false
end
```

Comprueba si la regla especificada existe entre las disponibles para la persona que llama. EventBridge

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts "Rule found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.rules.count.positive?
        if rule_found?(response.rules, rule_name)
          puts "Rule found."
          return true
```

```
        end
      end
    end
  end
  puts "Rule not found."
  return false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  return false
end
```

Crea una regla en EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
```

```
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        "aws.ec2"
      ],
      'detail-type': [
        "EC2 Instance State-change Notification"
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: "ENABLED",
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
```

```
      put_targets_response.failed_entry_count > 0
    puts "Error(s) adding target to rule:"
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    return false
  else
    return true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts "If the rule was created, you must add the target " \
    "to the rule yourself, or delete the rule yourself and try again."
  return false
end
```

Comprueba si el grupo de registros especificado existe entre los disponibles para la persona que llama en Amazon CloudWatch Logs.

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts "Log group found."
        return true
      end
```

```
    end
  end
  puts "Log group not found."
  return false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  return false
end
```

Cree un grupo de CloudWatch registros en Logs.

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts "Log group created."
  return true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  return false
end
```

Escribe un evento en una secuencia de CloudWatch registros en Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
```

```
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '495426724868310740095796045676567882148068632824696073EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  unless sequence_token.empty?
    event[:sequence_token] = sequence_token
  end

  response = cloudwatchlogs_client.put_log_events(event)
```

```
  puts "Message logged."
  return response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end
```

Reinicie una instancia de Amazon Elastic Compute Cloud (Amazon EC2) y añada información sobre la actividad relacionada a una secuencia de registros en Logs. CloudWatch

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
```

```
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ""

  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    "This might take a few minutes..."
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts "Instance stopped."
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts "Attempting to restart the instance. This might take a few minutes..."
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts "Instance restarted."
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  return true
rescue StandardError => e
  puts "Error creating log stream or stopping or restarting the instance: " \
    "#{e.message}"
  log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  return false
end
```

Muestra información sobre la actividad de una regla en EventBridge.

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts "Attempting to display rule activity..."
  response = cloudwatch_client.get_metric_statistics(
    namespace: "AWS/Events",
    metric_name: "Invocations",
    dimensions: [
      {
        name: "RuleName",
        value: rule_name
      }
    ],
```

```
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ["Sum"],
    unit: "Count"
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      "specified time period."
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Muestra la información de registro de todos los flujos de registros de un grupo de CloudWatch registros.

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts "Attempting to display log stream data for the log group " \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
```

```
    log_group_name: log_group_name,
    order_by: "LastEventTime",
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
      describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts "-" * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
          get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts "No log messages for this log stream."
      end
    end
  end
rescue StandardError => e
  puts "Error getting information about the log streams or their messages: " \
    "#{e.message}"
end
```

Muestra un recordatorio a la persona que llama para que limpie manualmente AWS los recursos asociados que ya no necesite.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
```

```
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts "-" * 10
  puts "Some of the following AWS resources might still exist in your account."
  puts "If you no longer want to use this code example, then to clean up"
  puts "your AWS account and avoid unexpected costs, you might want to"
  puts "manually delete any of the following resources if they exist:"
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end

# Example usage:
def run_me
  # Properties for the Amazon SNS topic.
  topic_name = "aws-doc-sdk-examples-topic"
  email_address = "mary@example.com"
  # Properties for the IAM role.
  role_name = "aws-doc-sdk-examples-cloudwatch-events-rule-role"
  # Properties for the Amazon EventBridge rule.
  rule_name = "aws-doc-sdk-examples-ec2-state-change"
  rule_description = "Triggers when any available EC2 instance starts."
  instance_state = "running"
  target_id = "sns-topic"
  # Properties for the Amazon EC2 instance.
  instance_id = "i-033c48ef067af3dEX"
  # Properties for displaying the event rule's activity.
  start_time = Time.now - 600 # Go back over the past 10 minutes
                              # (10 minutes * 60 seconds = 600 seconds).
  end_time = Time.now
  period = 60 # Look back every 60 seconds over the past 10 minutes.
  # Properties for the Amazon CloudWatch Logs log group.
  log_group_name = "aws-doc-sdk-examples-cloudwatch-log"
  # AWS service clients for this code example.
  region = "us-east-1"
```

```
sts_client = Aws::STS::Client.new(region: region)
sns_client = Aws::SNS::Client.new(region: region)
iam_client = Aws::IAM::Client.new(region: region)
cloudwatchevents_client = Aws::CloudWatchEvents::Client.new(region: region)
ec2_client = Aws::EC2::Client.new(region: region)
cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
cloudwatchlogs_client = Aws::CloudWatchLogs::Client.new(region: region)

# Get the caller's account ID for use in forming
# Amazon Resource Names (ARNs) that this code relies on later.
account_id = sts_client.get_caller_identity.account

# If the Amazon SNS topic doesn't exist, create it.
topic_arn = "arn:aws:sns:#{region}:#{account_id}:#{topic_name}"
unless topic_exists?(sns_client, topic_arn)
  topic_arn = create_topic(sns_client, topic_name, email_address)
  if topic_arn == "Error"
    puts "Could not create the Amazon SNS topic correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
    exit 1
  end
end

# If the IAM role doesn't exist, create it.
role_arn = "arn:aws:iam::#{account_id}:role/#{role_name}"
unless role_exists?(iam_client, role_arn)
  role_arn = create_role(iam_client, role_name)
  if role_arn == "Error"
    puts "Could not create the IAM role correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon EventBridge rule doesn't exist, create it.
unless rule_exists?(cloudwatchevents_client, rule_name)
  unless rule_created?(
    cloudwatchevents_client,
    rule_name,
    rule_description,
    instance_state,
```

```
    role_arn,
    target_id,
    topic_arn
  )
    puts "Could not create the Amazon EventBridge rule correctly. " \
      "Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
unless log_group_exists?(cloudwatchlogs_client, log_group_name)
  unless log_group_created?(cloudwatchlogs_client, log_group_name)
    puts "Could not create the Amazon CloudWatch Logs log group " \
    "correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# Restart the Amazon EC2 instance, which triggers the rule.
unless instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  puts "Could not restart the instance to trigger the rule. " \
    "Continuing anyway to show information about the rule and logs..."
end

# Display how many times the rule was triggered over the past 10 minutes.
display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)

# Display related log data in Amazon CloudWatch Logs.
```

```
  display_log_data(cloudwatchlogs_client, log_group_name)

  # Reminder the caller to clean up any AWS resources that are used
  # by this code example and are no longer needed.
  manual_cleanup_notice(
    topic_name, role_name, rule_name, log_group_name, instance_id
  )
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener detalles de la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Ruby .
  - PutEvents
  - PutRule

AWS Glue ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with AWS Glue.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones
- Escenarios

Acciones

Creación de un rastreador

El siguiente ejemplo de código muestra cómo crear un AWS Glue rastreador.

SDK para Ruby

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
 crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
```

```
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta CreateCrawlerla Referencia AWS SDK for Ruby de la API.

Creación de una definición de trabajo

El siguiente ejemplo de código muestra cómo crear una definición de AWS Glue trabajo.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new job with the specified configuration.
```

```
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: "glueetl",
        script_location: script_location,
        python_version: "3"
      },
      glue_version: "3.0"
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta CreateJobla Referencia AWS SDK for Ruby de la API.

Eliminación de un rastreador

El siguiente ejemplo de código muestra cómo eliminar un AWS Glue rastreador.

SDK para Ruby

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta DeleteCrawlerla Referencia AWS SDK for Ruby de la API.

Eliminación de una base de datos del Catálogo de datos

En el siguiente ejemplo de código se muestra cómo eliminar una base de datos del AWS Glue Data Catalog.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta DeleteDatabasela Referencia AWS SDK for Ruby de la API.

Eliminación de una definición de trabajo

El siguiente ejemplo de código muestra cómo eliminar una definición de AWS Glue trabajo y todas las ejecuciones asociadas.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta DeleteJobla Referencia AWS SDK for Ruby de la API.

Eliminación de una tabla de una base de datos

El siguiente ejemplo de código muestra cómo eliminar una tabla de una AWS Glue Data Catalog base de datos.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta DeleteTablela Referencia AWS SDK for Ruby de la API.

Obtención de un rastreador

El siguiente ejemplo de código muestra cómo obtener un AWS Glue rastreador.

SDK para Ruby

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil if
 not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta GetCrawlerla Referencia AWS SDK for Ruby de la API.

Obtención de una base de datos del Catálogo de datos

En el siguiente ejemplo de código se muestra cómo obtener una base de datos del AWS Glue Data Catalog.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
 if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get database #{name}: \n#{e.message}")
  raise
  end
```

- Para obtener más información sobre la API, consulta GetDatabasela Referencia AWS SDK for Ruby de la API.

Obtención de una ejecución de trabajo

El siguiente ejemplo de código muestra cómo ejecutar un AWS Glue trabajo.

SDK para Ruby

> **Note**
>
> Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta GetJobRunla Referencia AWS SDK for Ruby de la API.

Obtención de ejecuciones de un trabajo

El siguiente ejemplo de código muestra cómo obtener las ejecuciones de un AWS Glue trabajo.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta GetJobRunsla Referencia AWS SDK for Ruby de la API.

Obtención de obtener tablas de una base de datos

En el siguiente ejemplo de código se muestra cómo obtener tablas de una base de datos del AWS Glue Data Catalog.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta GetTablesla Referencia AWS SDK for Ruby de la API.

Enumeración de las definiciones de trabajos

El siguiente ejemplo de código muestra cómo enumerar las definiciones de AWS Glue trabajo.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for Ruby de la API.

Inicio de un rastreador

El siguiente ejemplo de código muestra cómo iniciar un AWS Glue rastreador.

SDK para Ruby

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta StartCrawlerla Referencia AWS SDK for Ruby de la API.

Inicio de una ejecución de trabajo

El siguiente ejemplo de código muestra cómo iniciar la ejecución de un AWS Glue trabajo.

SDK para Ruby

> **Note**
>
> Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
```

```
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta StartJobRunla Referencia AWS SDK for Ruby de la API.

Escenarios

Comenzar a ejecutar rastreadores y trabajos

En el siguiente ejemplo de código, se muestra cómo:

- Crear un rastreador que rastree un bucket de Amazon S3 público y generar una base de datos de metadatos con formato CSV.

- Enumera información sobre bases de datos y tablas en tu AWS Glue Data Catalog.
- Crear un trabajo para extraer datos CSV del bucket de S3, transformar los datos y cargar el resultado con formato JSON en otro bucket de S3.
- Incluir información sobre las ejecuciones de trabajos, ver algunos de los datos transformados y limpiar los recursos.

Para obtener más información, consulte el tutorial: Primeros pasos con AWS Glue Studio.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree una clase que agrupe AWS Glue las funciones utilizadas en el escenario.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil if
 not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
```

```
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end
```

```
  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
 if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get database #{name}: \n#{e.message}")
  raise
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
```

```
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: "glueetl",
        script_location: script_location,
        python_version: "3"
      },
      glue_version: "3.0"
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
```

```
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
```

```
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
 file to.
  # @return [void]
  def upload_job_script(file_path, bucket_resource)
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
       body: file,
       bucket: bucket_resource.name,
       key: file_path
     })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end

end
```

Crear una clase que ejecute el escenario.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
```

```
  end

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)

    new_step(1, "Create a crawler")
    puts "Checking for crawler #{crawler_name}."
    crawler = wrapper.get_crawler(crawler_name)
    if crawler == false
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
db_prefix, data_source)
      puts "Successfully created #{crawler_name}:"
      crawler = wrapper.get_crawler(crawler_name)
      puts JSON.pretty_generate(crawler).yellow
    end
    print "\nDone!\n".green

    new_step(2, "Run a crawler to output a database.")
    puts "Location of input data analyzed by crawler: #{data_source}"
    puts "Outputs: a Data Catalog database in CSV format containing metadata on
input."
    wrapper.start_crawler(crawler_name)
    puts "Starting crawler... (this typically takes a few minutes)"
    crawler_state = nil
    while crawler_state != "READY"
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]["state"]
      print "Status check: #{crawler_state}.".yellow
    end
    print "\nDone!\n".green

    new_step(3, "Query the database.")
    database = wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    print "#{database}".yellow
    puts "\nThe database contains these tables:"
    tables = wrapper.get_tables(db_name)
    tables.each_with_index do |table, index|
      print "\t#{index + 1}. #{table['name']}".yellow
    end
    print "\nDone!\n".green
```

```
    new_step(4, "Create a job definition that runs an ETL script.")
    puts "Uploading Python ETL script to S3..."
    wrapper.upload_job_script(job_script, @glue_bucket)
    puts "Creating job definition #{job_name}:\n"
    response = wrapper.create_job(job_name, "Getting started example job.",
@glue_service_role.arn, "s3://#{@glue_bucket.name}/#{job_script}")
    puts JSON.pretty_generate(response).yellow
    print "\nDone!\n".green

    new_step(5, "Start a new job")
    job_run_status = nil
    job_run_id = wrapper.start_job_run(
      job_name,
      db_name,
      tables[0]["name"],
      @glue_bucket.name
    )
    puts "Job #{job_name} started. Let's wait for it to run."
    until ["SUCCEEDED", "STOPPED", "FAILED", "TIMEOUT"].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]["job_run_state"]
      print "Status check: #{job_name}/#{job_run_id} - #{job_run_status}.".yellow
    end
    print "\nDone!\n".green

    new_step(6, "View results from a successful job run.")
    if job_run_status == "SUCCEEDED"
      puts "Data from your job run is stored in your S3 bucket
'#{@glue_bucket.name}'. Files include:"
      begin

        # Print the key name of each object in the bucket.
        @glue_bucket.objects.each do |object_summary|
          if object_summary.key.include?("run-")
            print "#{object_summary.key}".yellow
          end
        end

        # Print the first 256 bytes of a run file
        desired_sample_objects = 1
        @glue_bucket.objects.each do |object_summary|
          if object_summary.key.include?("run-")
            if desired_sample_objects > 0
```

```
              sample_object = @glue_bucket.object(object_summary.key)
              sample = sample_object.get(range: "bytes=0-255").body.read
              puts "\nSample run file contents:"
              print "#{sample}".yellow
              desired_sample_objects -= 1
            end
          end
        end
      rescue Aws::S3::Errors::ServiceError => e
        logger.error(
          "Couldn't get job run data. Here's why: %s: %s",
          e.response.error.code, e.response.error.message
        )
        raise
      end
    end
    print "\nDone!\n".green

    new_step(7, "Delete job definition and crawler.")
    wrapper.delete_job(job_name)
    puts "Job deleted: #{job_name}."
    wrapper.delete_crawler(crawler_name)
    puts "Crawler deleted: #{crawler_name}."
    wrapper.delete_table(db_name, tables[0]["name"])
    puts "Table deleted: #{tables[0]["name"]} in #{db_name}."
    wrapper.delete_database(db_name)
    puts "Database deleted: #{db_name}."
    print "\nDone!\n".green
  end
end

def main

  banner("../../helpers/banner.txt")
  puts
 "#########################################################################################
  puts "#
                          #".yellow
  puts "#                                          EXAMPLE CODE DEMO:
                          #".yellow
  puts "#                                              AWS Glue
                          #".yellow
  puts "#
                          #".yellow
```

```
 puts
"#########################################################################################
 puts ""
 puts "You have launched a demo of AWS Glue using the AWS for Ruby v3 SDK. Over the
next 60 seconds, it will"
 puts "do the following:"
 puts "    1. Create a crawler."
 puts "    2. Run a crawler to output a database."
 puts "    3. Query the database."
 puts "    4. Create a job definition that runs an ETL script."
 puts "    5. Start a new job."
 puts "    6. View results from a successful job run."
 puts "    7. Delete job definition and crawler."
 puts ""

 confirm_begin
 billing
 security
 puts "\e[H\e[2J"

 # Set input file names
 job_script_filepath = "job_script.py"
 resource_names = YAML.load_file("resource_names.yaml")

 # Instantiate existing IAM role.
 iam = Aws::IAM::Resource.new(region: "us-east-1")
 iam_role_name = resource_names["glue_service_role"]
 iam_role = iam.role(iam_role_name)

 # Instantiate existing S3 bucket.
 s3 = Aws::S3::Resource.new(region: "us-east-1")
 s3_bucket_name = resource_names["glue_bucket"]
 s3_bucket = s3.bucket(s3_bucket_name)

 scenario = GlueCrawlerJobScenario.new(
   Aws::Glue::Client.new(region: "us-east-1"),
   iam_role,
   s3_bucket,
   @logger
 )

 random_int = rand(10 ** 4)
 scenario.run(
   "doc-example-crawler-#{random_int}",
```

```
    "doc-example-database-#{random_int}",
    "doc-example-#{random_int}-",
    "s3://crawler-public-us-east-1/flight/2016/csv",
    job_script_filepath,
    "doc-example-job-#{random_int}"
  )

  puts "-" * 88
  puts "You have reached the end of this tour of AWS Glue."
  puts "To destroy CDK-created resources, run:\n       cdk destroy"
  puts "-" * 88

end
```

Cree un script ETL que sirva AWS Glue para extraer, transformar y cargar datos durante la ejecución de los trabajos.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the data
 to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
```

```
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Para obtener detalles de la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Ruby .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Ejemplos de IAM con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Ruby IAM.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones
- Escenarios

Acciones

Asociación de una política a un rol

En el siguiente ejemplo de código, se muestra cómo asociar una política de IAM a un rol.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
```

```
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
```

```
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta AttachRolePolicyla Referencia AWS SDK for Ruby de la API.

Asociar una política a un usuario

El siguiente ejemplo de código muestra cómo adjuntar una política de IAM a un usuario.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Attaches a policy to a user
#
# @param user_name [String] The name of the user
# @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
# @return [Boolean] true if successful, false otherwise
def attach_policy_to_user(user_name, policy_arn)
  @iam_client.attach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error attaching policy to user: #{e.message}")
  false
end
```

- Para obtener más información sobre la API, consulta AttachUserPolicyla Referencia AWS SDK for Ruby de la API.

Crear una política.

En el siguiente ejemplo de código, se muestra cómo crear una política de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
 #{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
```

```
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta CreatePolicyla Referencia AWS SDK for Ruby de la API.

Crear un rol

En el siguiente ejemplo de código, se muestra cómo crear un rol de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # Creates a role and attaches policies to it.
 #
 # @param role_name [String] The name of the role.
 # @param assume_role_policy_document [Hash] The trust relationship policy
document.
 # @param policy_arns [Array<String>] The ARNs of the policies to attach.
 # @return [String, nil] The ARN of the new role if successful, or nil if an error
occurred.
 def create_role(role_name, assume_role_policy_document, policy_arns)
   response = @iam_client.create_role(
     role_name: role_name,
     assume_role_policy_document: assume_role_policy_document.to_json
   )
   role_arn = response.role.arn

   policy_arns.each do |policy_arn|
     @iam_client.attach_role_policy(
       role_name: role_name,
       policy_arn: policy_arn
     )
```

```
    end

    role_arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating role: #{e.message}")
    nil
  end
```

- Para obtener más información sobre la API, consulta CreateRolela Referencia AWS SDK for Ruby de la API.

Creación de un rol vinculado al servicio

En el siguiente ejemplo de código se muestra cómo crear un rol vinculado al servicio de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # Creates a service-linked role
  #
  # @param service_name [String] The service name to create the role for.
  # @param description [String] The description of the service-linked role.
  # @param suffix [String] Suffix for customizing role name.
  # @return [String] The name of the created role
  def create_service_linked_role(service_name, description, suffix)
    response = @iam_client.create_service_linked_role(
      aws_service_name: service_name, description: description, custom_suffix:
suffix,)
    role_name = response.role.role_name
    @logger.info("Created service-linked role #{role_name}.")
    role_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't create service-linked role for #{service_name}. Here's
why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
```

```
end
```

- Para obtener más información sobre la API, consulta CreateServiceLinkedRolela Referencia AWS SDK for Ruby de la API.

Creación de un usuario

En el siguiente ejemplo de código, se muestra cómo crear un usuario de IAM.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Creates a user and their login profile
#
# @param user_name [String] The name of the user
# @param initial_password [String] The initial password for the user
# @return [String, nil] The ID of the user if created, or nil if an error occurred
def create_user(user_name, initial_password)
  response = @iam_client.create_user(user_name: user_name)
  @iam_client.wait_until(:user_exists, user_name: user_name)
  @iam_client.create_login_profile(
    user_name: user_name,
    password: initial_password,
    password_reset_required: true
  )
  @logger.info("User '#{user_name}' created successfully.")
  response.user.user_id
```

```
  rescue Aws::IAM::Errors::EntityAlreadyExists
    @logger.error("Error creating user '#{user_name}': user already exists.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating user '#{user_name}': #{e.message}")
    nil
  end
```

- Para obtener más información sobre la API, consulta CreateUserla Referencia AWS SDK for Ruby de la API.

Creación de una clave de acceso

En el siguiente ejemplo de código, se muestra cómo crear una clave de acceso de IAM.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este módulo de ejemplo muestra, crea, desactiva y elimina las claves de acceso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end
```

```
  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded => e
    @logger.error("Error creating access key: limit exceeded. Cannot create more.")
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
```

```
      access_key_id: access_key_id,
      status: "Inactive"
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta CreateAccessKeyla Referencia AWS SDK for Ruby de la API.

Crear un alias para una cuenta

El siguiente ejemplo de código muestra cómo crear un alias para una cuenta de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, cree y elimine los alias de la cuenta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
```

```
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta CreateAccountAliasla Referencia AWS SDK for Ruby de la API.

Crear una política insertada para un usuario

El siguiente ejemplo de código muestra cómo crear una política de IAM insertada para un usuario.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Creates an inline policy for a specified user.
# @param username [String] The name of the IAM user.
# @param policy_name [String] The name of the policy to create.
# @param policy_document [String] The JSON policy document.
# @return [Boolean]
def create_user_policy(username, policy_name, policy_document)
  @iam_client.put_user_policy({
    user_name: username,
    policy_name: policy_name,
    policy_document: policy_document
```

```
    })
    @logger.info("Policy #{policy_name} created for user #{username}.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't create policy #{policy_name} for user #{username}.
 Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    false
  end
```

- Para obtener más información sobre la API, consulta PutUserPolicyla Referencia AWS SDK for Ruby de la API.

Eliminación de un rol

En el siguiente ejemplo de código, se muestra cómo eliminar un rol de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # Deletes a role and its attached policies.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    begin
      # Detach and delete attached policies
      @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
        response.attached_policies.each do |policy|
          @iam_client.detach_role_policy({
                                           role_name: role_name,
                                           policy_arn: policy.policy_arn
                                         })
          # Check if the policy is a customer managed policy (not AWS managed)
          unless policy.policy_arn.include?("aws:policy/")
            @iam_client.delete_policy({ policy_arn: policy.policy_arn })
```

```
            @logger.info("Deleted customer managed policy #{policy.policy_name}.")
          end
        end
      end

      # Delete the role
      @iam_client.delete_role({ role_name: role_name })
      @logger.info("Deleted role #{role_name}.")
    rescue Aws::IAM::Errors::ServiceError => e
      @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
 why:")
      @logger.error("\t#{e.code}: #{e.message}")
      raise
    end
  end
```

- Para obtener más información sobre la API, consulta DeleteRolela Referencia AWS SDK for Ruby de la API.

Eliminar un certificado de servidor

El siguiente ejemplo de código muestra cómo eliminar un certificado de servidor de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, actualice y elimine certificados del servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
```

```
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
```

```
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta DeleteServerCertificatela Referencia AWS SDK for Ruby de la API.

Eliminar un rol vinculado a un servicio

El siguiente ejemplo de código muestra cómo eliminar un rol de IAM vinculado a un servicio.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # Deletes a service-linked role.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_service_linked_role(role_name)
    response = @iam_client.delete_service_linked_role(role_name: role_name)
    task_id = response.deletion_task_id
    check_deletion_status(role_name, task_id)
  rescue Aws::Errors::ServiceError => e
    handle_deletion_error(e, role_name)
  end

  private
```

```
# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id)
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)
    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
def handle_deletion_error(e, role_name)
  unless e.code == "NoSuchEntity"
    @logger.error("Couldn't delete #{role_name}. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Para obtener más información sobre la API, consulta DeleteServiceLinkedRolela Referencia AWS SDK for Ruby de la API.

Eliminación de un usuario

En el siguiente ejemplo de código, se muestra cómo eliminar un usuario de IAM.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
 end
```

- Para obtener más información sobre la API, consulta DeleteUserla Referencia AWS SDK for Ruby de la API.

Eliminación de una clave de acceso

En el siguiente ejemplo de código, se muestra cómo eliminar una clave de acceso de IAM.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este módulo de ejemplo muestra, crea, desactiva y elimina las claves de acceso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
```

```
    @logger.info("Access key created for user '#{user_name}':
 #{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded => e
    @logger.error("Error creating access key: limit exceeded. Cannot create more.")
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: "Inactive"
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta DeleteAccessKeyla Referencia AWS SDK for Ruby de la API.

Eliminar un alias de cuenta

El siguiente ejemplo de código muestra cómo eliminar un alias de cuenta de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, cree y elimine los alias de la cuenta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
```

```
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta DeleteAccountAliasla Referencia AWS SDK for Ruby de la API.

Eliminación de una política insertada de un usuario

En el siguiente ejemplo de código se muestra cómo eliminar una política de IAM insertada de un usuario.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
 end
```

- Para obtener más información sobre la API, consulta DeleteUserPolicyla Referencia AWS SDK for Ruby de la API.

Desasociación de una política de un rol

En el siguiente ejemplo de código, se muestra cómo desasociar una política de IAM de un rol.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
 #{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
```

```
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta DetachRolePolicyla Referencia AWS SDK for Ruby de la API.

Desasociar una política de un usuario

El siguiente ejemplo de código muestra cómo desadjuntar una política de IAM de un usuario.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Detaches a policy from a user
#
# @param user_name [String] The name of the user
# @param policy_arn [String] The ARN of the policy to detach
# @return [Boolean] true if the policy was successfully detached, false otherwise
def detach_user_policy(user_name, policy_arn)
  @iam_client.detach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
```

```
    @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
successfully.")
    true
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error detaching policy: Policy or user does not exist.")
    false
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from user '#{user_name}': #{e.message}")
    false
  end
```

- Para obtener más información sobre la API, consulta DetachUserPolicyla Referencia AWS SDK for Ruby de la API.

Obtención de una política

En el siguiente ejemplo de código se muestra cómo obtener una política de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
```

```
    raise
  end
```

- Para obtener más información sobre la API, consulta GetPolicyla Referencia AWS SDK for Ruby de la API.

Obtención de un rol

En el siguiente ejemplo de código se muestra cómo obtener un rol de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                               role_name: name,
                             }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- Para obtener más información sobre la API, consulta GetRolela Referencia AWS SDK for Ruby de la API.

Obtención de un usuario

El siguiente ejemplo de código muestra cómo obtener un usuario de IAM.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # Retrieves a user's details
 #
 # @param user_name [String] The name of the user to retrieve
 # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
 def get_user(user_name)
   response = @iam_client.get_user(user_name: user_name)
   response.user
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("User '#{user_name}' not found.")
   nil
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error retrieving user '#{user_name}': #{e.message}")
   nil
 end
```

- Para obtener más información sobre la API, consulta GetUserla Referencia AWS SDK for Ruby de la API.

Obtención de la política de contraseñas de la cuenta

En el siguiente ejemplo de código se muestra cómo obtener la política de contraseñas de la cuenta de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "IAMPolicyManager"
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
    begin
      response = @iam_client.get_account_password_policy
      @logger.info("The account password policy is:
 #{response.password_policy.to_h}")
    rescue Aws::IAM::Errors::NoSuchEntity
      @logger.info("The account does not have a password policy.")
    rescue Aws::Errors::ServiceError => e
      @logger.error("Couldn't print the account password policy. Error: #{e.code} -
 #{e.message}")
      raise
    end
  end
end
```

- Para obtener más información sobre la API, consulta GetAccountPasswordPolicyla Referencia AWS SDK for Ruby de la API.

Enumeración de proveedores de SAML

En el siguiente ejemplo de código se muestra cómo enumerar proveedores SAML de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Para obtener información acerca de la API, consulte ListSAMLProviders en la referencia de la API de AWS SDK for Ruby .

Enumerar las claves de acceso de un usuario

En el siguiente ejemplo de código se muestra cómo enumerar las claves de acceso de IAM de un usuario.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este módulo de ejemplo muestra, crea, desactiva y elimina las claves de acceso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
```

```
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded => e
    @logger.error("Error creating access key: limit exceeded. Cannot create more.")
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: "Inactive"
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
```

```
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta ListAccessKeysla Referencia AWS SDK for Ruby de la API.

Mostrar alias de cuenta

El siguiente ejemplo de código muestra cómo enumerar los alias de cuentas de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, cree y elimine los alias de la cuenta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end
```

```
  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta ListAccountAliasesla Referencia AWS SDK for Ruby de la API.

Enumeración de grupos

En el siguiente ejemplo de código se muestra cómo enumerar grupos de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Para obtener más información sobre la API, consulta ListGroupsla Referencia AWS SDK for Ruby de la API.

Enumeración de políticas insertadas para un rol

En el siguiente ejemplo de código se muestra cómo enumerar las políticas insertadas de un rol de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Lists policy ARNs attached to a role
#
# @param role_name [String] The name of the role
# @return [Array<String>] List of policy ARNs
def list_attached_policy_arns(role_name)
  response = @iam_client.list_attached_role_policies(role_name: role_name)
  response.attached_policies.map(&:policy_arn)
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing policies attached to role: #{e.message}")
  []
end
```

- Para obtener más información sobre la API, consulta ListRolePoliciesla Referencia AWS SDK for Ruby de la API.

Enumeración de políticas

En el siguiente ejemplo de código se muestra cómo enumerar políticas de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
 #{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
```

```
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta ListPolicieslla Referencia AWS SDK for Ruby de la API.

Enumeración de las políticas asociadas a un rol

En el siguiente ejemplo de código se muestra cómo enumerar las políticas asociadas a un rol de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
```

```
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
```

```
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta ListAttachedRolePoliciesla Referencia AWS SDK for Ruby de la API.

Enumeración de roles

En el siguiente ejemplo de código se muestra cómo enumerar roles de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count
      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end

  role_names
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't list roles for the account. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- Para obtener más información sobre la API, consulta ListRoleslа Referencia AWS SDK for Ruby de la API.

Elaborar listas de certificados de servidor

El siguiente ejemplo de código muestra cómo enumerar certificados de servidor de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, actualice y elimine certificados del servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end
```

```
    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta ListServerCertificatesla Referencia AWS SDK for Ruby de la API.

Enumeración de usuarios

En el siguiente ejemplo de código se muestra cómo enumerar usuarios de IAM.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing users: #{e.message}")
  []
end
```

- Para obtener más información sobre la API, consulta ListUsersla Referencia AWS SDK for Ruby de la API.

Actualizar un certificado de servidor

En el siguiente ejemplo de código, se muestra cómo actualizar un certificado de servidor de IAM.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, actualice y elimine certificados del servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end
```

```
    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obtener más información sobre la API, consulta UpdateServerCertificatela Referencia AWS SDK for Ruby de la API.

Actualizar un usuario

El siguiente ejemplo de código muestra cómo actualizar un usuario de IAM.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # Updates an IAM user's name
 #
 # @param current_name [String] The current name of the user
 # @param new_name [String] The new name of the user
 def update_user_name(current_name, new_name)
   @iam_client.update_user(user_name: current_name, new_user_name: new_name)
   true
 rescue StandardError => e
   @logger.error("Error updating user name from '#{current_name}' to '#{new_name}':
#{e.message}")
   false
 end
```

- Para obtener más información sobre la API, consulta UpdateUserla Referencia AWS SDK for Ruby de la API.

Escenarios

Crear un usuario y asumir un rol

En el siguiente ejemplo de código, se muestra cómo crear un usuario y asumir un rol.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

- Crear un usuario que no tenga permisos.
- Crear un rol que conceda permiso para enumerar los buckets de Amazon S3 para la cuenta.
- Agregar una política para que el usuario asuma el rol.
- Asumir el rol y enumerar los buckets de S3 con credenciales temporales, y después limpiar los recursos.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree un usuario de IAM y un rol que conceda permiso para enumerar los buckets de Amazon S3. El usuario solo tiene derechos para asumir el rol. Después de asumir el rol, use las credenciales temporales para enumerar los buckets de la cuenta.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
```

```
    puts("Give AWS time to propagate resources...")
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Tried and failed to create demo user.")
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name: user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: "2012-10-17",
```

```
      Statement: [{
                    Effect: "Allow",
                    Principal: {'AWS': user.arn},
                    Action: "sts:AssumeRole"
                  }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account, and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: "2012-10-17",
      Statement: [{
                    Effect: "Allow",
                    Action: "s3:ListAllMyBuckets",
                    Resource: "arn:aws:s3:::*"
                  }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
```

```
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create a policy and attach it to role #{role.role_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Creates an inline policy for a user that lets the user assume a role.
 #
 # @param policy_name [String] The name to give the policy.
 # @param user [Aws::IAM::User] The user that owns the policy.
 # @param role [Aws::IAM::Role] The role that can be assumed.
 # @return [Aws::IAM::UserPolicy] The newly created policy.
 def create_user_policy(policy_name, user, role)
   policy_document = {
     Version: "2012-10-17",
     Statement: [{
                   Effect: "Allow",
                   Action: "sts:AssumeRole",
                   Resource: role.arn
                 }]
   }.to_json
   @iam_client.put_user_policy(
     user_name: user.user_name,
     policy_name: policy_name,
     policy_document: policy_document
   )
   puts("Created an inline policy for #{user.user_name} that lets the user assume
role #{role.role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Creates an Amazon S3 resource with specified credentials. This is separated into
a
 # factory function so that it can be mocked for unit testing.
 #
 # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
```

```
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
   s3_resource.buckets.each do |bucket|
     @logger.info "\t#{bucket.name}"
     count -= 1
     break if count.zero?
   end
 rescue Aws::Errors::ServiceError => e
   if e.code == "AccessDenied"
     puts("Attempt to list buckets with no permissions: AccessDenied.")
   else
     @logger.info("Couldn't list buckets for the account. Here's why: ")
     @logger.info("\t#{e.code}: #{e.message}")
     raise
   end
 end

 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
```

```
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end

 # Deletes a role. If the role has policies attached, they are detached and
 # deleted before the role is deleted.
 #
 # @param role_name [String] The name of the role to delete.
 def delete_role(role_name)
   @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
     @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
     @iam_client.delete_policy(policy_arn: policy.policy_arn)
     @logger.info("Detached and deleted policy #{policy.policy_name}.")
   end
   @iam_client.delete_role({ role_name: role_name })
   @logger.info("Role deleted: #{role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Deletes a user. If the user has inline policies or access keys, they are deleted
 # before the user is deleted.
 #
 # @param user [Aws::IAM::User] The user to delete.
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end
```

```
    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
 role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key))
  puts("Here are your buckets:")
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Ruby .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ejemplos de Kinesis con SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK for Ruby uso de Kinesis.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Ejemplos sin servidor

Ejemplos sin servidor

Invocar una función de Lambda desde un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir registros de un flujo de Kinesis. La función recupera la carga útil de Kinesis, la decodifica desde Base64 y registra el contenido del registro.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumir un evento de Kinesis con Lambda mediante Ruby.

```
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de un flujo de Kinesis. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Cómo informar de errores en los artículos de lote de Kinesis con Lambda mediante Ruby.

```
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
```

```
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
end
```

AWS KMS ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with AWS KMS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Crear una clave

El siguiente ejemplo de código muestra cómo crear un AWS KMS key.

SDK para Ruby

> **Note**
>
> Hay más información sobre GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-kms" # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: "CreatedBy",
                               tag_value: "ExampleUser"
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

- Para obtener más información sobre la API, consulta CreateKeyla Referencia AWS SDK for Ruby de la API.

Descifrar texto cifrado

En el siguiente ejemplo de código se muestra cómo descifrar texto cifrado que se cifró con una clave de KMS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-kms" # v2: require 'aws-sdk'
```

```
# Decrypted blob

blob =
 "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596
blob_packed = [blob].pack("H*")

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts "Raw text: "
puts resp.plaintext
```

- Para obtener información acerca de la API, consulte Decrypt en la referencia de la API de AWS SDK for Ruby .

Cifrar texto mediante una clave

En el siguiente ejemplo de código se muestra cómo cifrar texto utilizando una clave de KMS.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-kms"  # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
text = "1234567890"

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.encrypt({
  key_id: keyId,
  plaintext: text,
})

# Display a readable version of the resulting encrypted blob.
puts "Blob:"
puts resp.ciphertext_blob.unpack("H*")
```

- Para obtener más información acerca de la API, consulte Encrypt en la referencia de la API de AWS SDK for Ruby .

Pasar texto cifrado de una clave a otra

En el siguiente ejemplo de código se muestra cómo volver a cifrar texto cifrado de una clave KMS a otra.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-kms"  # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
 "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596
sourceCiphertextBlob = [blob].pack("H*")

# Replace the fictitious key ARN with a valid key ID
```

```
destinationKeyId = "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.re_encrypt({
  ciphertext_blob: sourceCiphertextBlob,
  destination_key_id: destinationKeyId
})

# Display a readable version of the resulting re-encrypted blob.
puts "Blob:"
puts resp.ciphertext_blob.unpack("H*")
```

- Para obtener más información sobre la API, consulta ReEncryptla Referencia AWS SDK for Ruby de la API.

Ejemplos de Lambda con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Lambda.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones
- Escenarios
- Ejemplos sin servidor

Acciones

Crear una función

En el siguiente ejemplo de código se muestra cómo crear una función de Lambda.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function. This
  #                      must include the file name and the function name.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
  #                            code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: "ruby2.7",
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
```

```
                                                    "LOG_LEVEL" => "info"
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name})
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- Para obtener más información sobre la API, consulta CreateFunctionla Referencia AWS SDK for Ruby de la API.

Eliminar una función

En el siguiente ejemplo de código se muestra cómo eliminar una función de Lambda.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deletes a Lambda function.
```

```
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name: function_name
    )
    print "Done!".green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
```

- Para obtener más información sobre la API, consulta DeleteFunctionla Referencia AWS SDK for Ruby de la API.

Obtener una función

En el ejemplo de código siguiente se muestra cómo obtener una función de Lambda.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
```

```
      {
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
```

- Para obtener más información sobre la API, consulta GetFunctionla Referencia AWS SDK for Ruby de la API.

Invocar una función

En el siguiente ejemplo de código se muestra cómo invocar una función de Lambda.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name}
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
```

```
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

- Para obtener información acerca de la API, consulte Invocar en la referencia de la API de AWS SDK for Ruby .

Mostrar funciones

En el ejemplo de código siguiente se muestra cómo enumerar funciones Lambda.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response["functions"].each do |function|
        functions.append(function["function_name"])
      end
    end
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

- Para obtener más información sobre la API, consulta ListFunctionsla Referencia AWS SDK for Ruby de la API.

Actualizar el código de la función

En el ejemplo de código siguiente se muestra cómo actualizar un código de una función de Lambda.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.

  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
 do |w|
```

```
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

- Para obtener más información sobre la API, consulta UpdateFunctionCodela Referencia AWS SDK for Ruby de la API.

Actualizar la configuración de la función

En el ejemplo de código siguiente se muestra cómo actualizar la configuración de una función de Lambda.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
```

```
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                   function_name: function_name,
                                                   environment: {
                                                     variables: {
                                                       "LOG_LEVEL" => log_level
                                                     }
                                                   }
                                                 })
    @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
 do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- Para obtener más información sobre la API, consulta UpdateFunctionConfigurationla Referencia AWS SDK for Ruby de la API.

Escenarios

Comenzar a usar las funciones

En el siguiente ejemplo de código, se muestra cómo:

- Crear un rol de IAM y una función de Lambda y, a continuación, cargar el código de controlador.
- Invocar la función con un único parámetro y obtener resultados.
- Actualizar el código de la función y configurar con una variable de entorno.
- Invocar la función con un nuevo parámetro y obtener resultados. Mostrar el registro de ejecución devuelto.
- Enumerar las funciones de su cuenta y, luego, limpiar los recursos.

Para obtener información, consulte Crear una función de Lambda con la consola.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Configurar los permisos de IAM de requisitos previos para que una función de Lambda pueda escribir registros.

```
  # Get an AWS Identity and Access Management (IAM) role.
  #
  # @param iam_role_name: The name of the role to retrieve.
  # @param action: Whether to create or destroy the IAM apparatus.
  # @return: The IAM role.
  def manage_iam(iam_role_name, action)
    role_policy = {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Effect': "Allow",
          'Principal': {
            'Service': "lambda.amazonaws.com"
          },
          'Action': "sts:AssumeRole"
        }
      ]
    }
    case action
    when "create"
      role = $iam_client.create_role(
        role_name: iam_role_name,
        assume_role_policy_document: role_policy.to_json
      )
      $iam_client.attach_role_policy(
        {
          policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
          role_name: iam_role_name
        }
      )
```

```
      $iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
        w.max_attempts = 5
        w.delay = 5
      end
      @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
      @logger.debug("Enforcing a 10-second sleep to allow IAM role to activate
 fully.")
      sleep(10)
      return role, role_policy.to_json
    when "destroy"
      $iam_client.detach_role_policy(
        {
          policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
          role_name: iam_role_name
        }
      )
      $iam_client.delete_role(
        role_name: iam_role_name
      )
      @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
    else
      raise "Incorrect action provided. Must provide 'create' or 'destroy'"
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating role or attaching policy:\n
 #{e.message}")
  end
```

Definir un controlador de Lambda que aumente un número dado como parámetro de invocación.

```
require "logger"

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
# @param event [Hash] Parameters sent when the function is invoked
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
  logger = Logger.new($stdout)
```

```
  log_level = ENV["LOG_LEVEL"]
  logger.level = case log_level
                 when "debug"
                   Logger::DEBUG
                 when "info"
                   Logger::INFO
                 else
                   Logger::ERROR
                 end
  logger.debug("This is a debug log message.")
  logger.info("This is an info log message. Code executed successfully!")
  number = event["number"].to_i
  incremented_number = number + 1
  logger.info("You provided #{number.round} and it was incremented to
 #{incremented_number.round}")
  incremented_number.round.to_s
end
```

Comprimir su función de Lambda en un paquete de implementación.

```
  # Creates a Lambda deployment package in .zip format.
  # This zip can be passed directly as a string to Lambda when creating the
 function.
  #
  # @param source_file: The name of the object, without suffix, for the Lambda file
 and zip.
  # @return: The deployment package.
  def create_deployment_package(source_file)
    Dir.chdir(File.dirname(__FILE__))
    if File.exist?("lambda_function.zip")
      File.delete("lambda_function.zip")
      @logger.debug("Deleting old zip: lambda_function.zip")
    end
    Zip::File.open("lambda_function.zip", create: true) {
      |zipfile|
      zipfile.add("lambda_function.rb", "#{source_file}.rb")
    }
    @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
    File.read("lambda_function.zip").to_s
  rescue StandardError => e
    @logger.error("There was an error creating deployment package:\n #{e.message}")
  end
```

Crear una nueva función de Lambda.

```
  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function. This
  #                      must include the file name and the function name.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
  #                            code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: "ruby2.7",
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    "LOG_LEVEL" => "info"
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name})
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Invocar su función de Lambda con parámetros de tiempo de ejecución opcionales.

```
  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name}
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

Actualizar la configuración de su función de Lambda para introducir una nueva variable de entorno.

```
  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                   function_name: function_name,
                                                   environment: {
                                                     variables: {
                                                       "LOG_LEVEL" => log_level
                                                     }
                                                   }
                                                 })
    @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
 do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Actualizar el código de su función de Lambda con un paquete de implementación diferente que contenga un código diferente.

```
  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.

  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
 do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

Enumerar todas las funciones de Lambda existentes mediante el paginador integrado.

```
  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response["functions"].each do |function|
        functions.append(function["function_name"])
      end
    end
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
```

```
end
```

Eliminar una función de Lambda específica.

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name: function_name
  )
  print "Done!".green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Ruby .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ejemplos sin servidor

Invocar una función de Lambda desde un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir registros de un flujo de Kinesis. La función recupera la carga útil de Kinesis, la decodifica desde Base64 y registra el contenido del registro.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumir un evento de Kinesis con Lambda mediante Ruby.

```
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Invocar una función de Lambda desde un desencadenador de Amazon SNS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Uso de un evento de SNS con Lambda mediante Ruby.

```
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Invocar una función de Lambda desde un desencadenador de Amazon SQS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de una cola de SQS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Uso de un evento de SQS con Lambda mediante Ruby.

```
def lambda_handler(event:, context:)
```

```
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de un flujo de Kinesis. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Cómo informar de errores en los artículos de lote de Kinesis con Lambda mediante Ruby.

```
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
```

```
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
end
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Amazon SQS.

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de una cola de SQS. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Notificación de los errores de los elementos del lote de SQS con Lambda mediante Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

Ejemplos de Amazon Polly con SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Amazon Polly.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Obtener las voces disponibles para su síntesis

En el siguiente ejemplo de código se muestra cómo obtener las voces de Amazon Polly disponibles para su síntesis.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-polly"  # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: "en-US")

  resp.voices.each do |v|
    puts v.name
    puts "  " + v.gender
    puts
  end
rescue StandardError => ex
  puts "Could not get voices"
  puts "Error message:"
  puts ex.message
end
```

- Para obtener más información sobre la API, consulta DescribeVoicesla Referencia AWS SDK for Ruby de la API.

Enumerar términos de pronunciación

En el siguiente ejemplo de código se muestra cómo enumerar términos de pronunciación de Amazon Polly.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-polly"  # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:" + l.attributes.alphabet
    puts "  Language:" + l.attributes.language
    puts
  end
rescue StandardError => ex
  puts "Could not get lexicons"
  puts "Error message:"
  puts ex.message
end
```

- Para obtener más información sobre la API, consulta ListLexiconsla Referencia AWS SDK for Ruby de la API.

Sintetizar voz a partir de texto

En el siguiente ejemplo de código se muestra cómo sintetizar voz a partir de texto con Amazon Polly.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-polly"  # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?
    puts "You must supply a filename"
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: " + filename
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
    output_format: "mp3",
    text: contents,
    voice_id: "Joanna",
  })
```

```
  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split(".")
  first_part = parts[0]
  mp3_file = first_part + ".mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: " + mp3_file
rescue StandardError => ex
  puts "Got error:"
  puts "Error message:"
  puts ex.message
end
```

- Para obtener más información sobre la API, consulta SynthesizeSpeechla Referencia AWS SDK for Ruby de la API.

Ejemplos de Amazon RDS con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Amazon RDS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Crear una instantánea de una instancia de base de datos

En el siguiente ejemplo de código se muestra cómo crear una instantánea de una instancia de base de datos de Amazon RDS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-rds"  # v2: require 'aws-sdk'

# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
                                db_snapshot_identifier: id
                              })
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- Para obtener información acerca de la API, consulte CreateDBSnapshot en la referencia de la API de AWS SDK for Ruby .

Describir instancias de base de datos

En el siguiente ejemplo de código se muestra cómo describir instancias de base de datos de Amazon RDS.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-rds"  # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
                          "name": i.id,
                          "status": i.db_instance_status
                        })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- Para obtener información acerca de la API, consulte DescribeDBInstances en la referencia de la API de AWS SDK for Ruby .

Describir grupos de parámetros de base de datos

En el siguiente ejemplo de código se muestra cómo describir grupos de parámetros de base de datos de Amazon RDS.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-rds"  # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Para obtener información detallada sobre la API, consulta DescribeDB ParameterGroups en la referencia de la AWS SDK for Ruby API.

Describir parámetros de un grupo de parámetros de base de datos

En el siguiente ejemplo de código se muestra cómo describir parámetros de un grupo de parámetros de base de datos de Amazon RDS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-rds"  # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Para obtener información sobre la API, consulte DescribeDBParameters en la Referencia de la API de AWS SDK for Ruby .

Describir instantáneas de instancias de base de datos

En el siguiente ejemplo de código se muestra cómo describir instantáneas de instancias de base de datos de Amazon RDS.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-rds" # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instance
# snapshots.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return instance_snapshots [Array, nil] All instance snapshots, or nil if error.
def list_instance_snapshots(rds_resource)
  instance_snapshots = []
  rds_resource.db_snapshots.each do |s|
    instance_snapshots.append({
                                "id": s.snapshot_id,
                                "status": s.status
                              })
  end
  instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
end
```

- Para obtener información acerca de la API, consulte DescribeDBSnapshots en la referencia de la API de AWS SDK for Ruby .

Ejemplos de Amazon S3 con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby mediante Amazon S3.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

Añadir reglas CORS a un bucket

En el siguiente ejemplo de código se muestra cómo añadir reglas de uso compartido de recursos entre orígenes (CORS) a un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
```

```
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obtener más información sobre la API, consulta PutBucketCorsla Referencia AWS SDK for Ruby de la API.

Añadir una política a un bucket

En el siguiente ejemplo de código se muestra cómo añadir una política a un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def set_policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obtener más información sobre la API, consulta PutBucketPolicyla Referencia AWS SDK for Ruby de la API.

Copiar un objeto de un bucket a otro

En el siguiente ejemplo de código se muestra cómo copiar un objeto de S3 de un bucket a otro.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Copie un objeto.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
```

```
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copie un objeto y añada cifrado del lado del servidor al objeto de destino.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                       copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
```

```
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta CopyObjectla Referencia AWS SDK for Ruby de la API.

Crear un bucket

En el siguiente ejemplo de código se muestra cómo crear un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
```

```
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta CreateBucketla Referencia AWS SDK for Ruby de la API.

Eliminación de reglas CORS de un bucket

En el siguiente ejemplo de código se muestra cómo eliminar reglas CORS de un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
```

```
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obtener más información sobre la API, consulta DeleteBucketCorsla Referencia AWS SDK for Ruby de la API.

Eliminar una política de un bucket

En el siguiente ejemplo de código se muestra cómo eliminar una política de un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
```

```
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obtener más información sobre la API, consulta DeleteBucketPolicyla Referencia AWS SDK for Ruby de la API.

Eliminar un bucket vacío

En el siguiente ejemplo de código se muestra cómo eliminar un bucket de S3 vacío.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
```

```
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Para obtener más información sobre la API, consulta DeleteBucketla Referencia AWS SDK for Ruby de la API.

Eliminar varios objetos

En el siguiente ejemplo de código se muestra cómo eliminar varios objetos de un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
  answer = gets.chomp.downcase
  if answer == "y"
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Para obtener más información sobre la API, consulta DeleteObjectsla Referencia AWS SDK for Ruby de la API.

Determinar la existencia y el tipo de contenido de un objeto

En los siguientes ejemplos de código se muestra cómo determinar la existencia y el tipo de contenido de un objeto en un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
 Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
```

```
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta HeadObjectla Referencia AWS SDK for Ruby de la API.

Obtener reglas CORS para un bucket

En el siguiente ejemplo de código se muestra cómo obtener reglas de uso compartido de recursos entre orígenes (CORS) para un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
 for the bucket.
  def get_cors
```

```
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- Para obtener más información sobre la API, consulta GetBucketCorsla Referencia AWS SDK for Ruby de la API.

Obtener un objeto de un bucket.

En el siguiente ejemplo de código se muestra cómo leer datos de un objeto en un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Obtenga un objeto.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is downloaded.
```

```
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Obtenga un objeto e informe de su estado de cifrado del lado del servidor.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
```

```
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta GetObjectla Referencia AWS SDK for Ruby de la API.

Obtener una política para un bucket

En el siguiente ejemplo de código se muestra cómo obtener la política para un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Wraps an Amazon S3 bucket policy.
```

```
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def get_policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- Para obtener más información sobre la API, consulta GetBucketPolicyla Referencia AWS SDK for Ruby de la API.

Obtener una lista de buckets

En el siguiente ejemplo de código se muestra cómo obtener una lista de buckets de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"
```

```
# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts "Found these buckets:"
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta ListBucketsla Referencia AWS SDK for Ruby de la API.

Obtener una lista de los objetos en un bucket

En el siguiente ejemplo de código se muestra cómo obtener una lista de los objetos en un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why: #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
```

```
  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta la ListObjectsversión 2 en la referencia de la AWS SDK for Ruby API.

Establecer la configuración de sitio web de un bucket

En el siguiente ejemplo de código se muestra cómo establecer la configuración de un sitio web para un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
```

```
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta PutBucketWebsitela Referencia AWS SDK for Ruby de la API.

Cargar un objeto en un bucket

En el siguiente ejemplo de código se muestra cómo cargar un objeto en un bucket de S3.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cargue un archivo con un cargador administrado (Object.upload_file).

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
```

```
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Cargue un archivo con Object.put.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success
```

```
  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Cargue un archivo con Object.put y añada cifrado del lado del servidor.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end
```

```
run_demo if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta PutObjectla Referencia AWS SDK for Ruby de la API.

Escenarios

Crear una URL prefirmada

En el siguiente ejemplo de código se muestra cómo crear una URL prefirmada para Amazon S3 y cargar un objeto.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"
require "net/http"

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:
 #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
```

```
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request("PUT", presigned_url.request_uri, object_content,
 "content_type" => "")
  end

  case response
  when Net::HTTPSuccess
    puts "Content uploaded!"
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Comenzar a usar buckets y objetos

En el siguiente ejemplo de código, se muestra cómo:

- Creación de un bucket y cargar un archivo en el bucket.
- Descargar un objeto desde un bucket.
- Copiar un objeto en una subcarpeta de un bucket.
- Obtención de una lista de los objetos de un bucket.
- Eliminación del bucket y todos los objetos que incluye.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "doc-example-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: "us-east-1" # Note: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Tried and failed to create demo bucket.")
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open("demo.txt", w) { |f| f.write("This is a demo file.") }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
 destination
```

```
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded file.
  def upload_file(bucket)
    File.open("demo.txt", "w+") { |f| f.write("This is a demo file.") }
    s3_object = bucket.object(File.basename("demo.txt"))
    s3_object.upload_file("demo.txt")
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      puts("Enter a name for the downloaded file: ")
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
```

```
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
```

```
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the Amazon S3 getting started demo!")
  puts("-" * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo!")
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME ==
 __FILE__
```

- Para obtener detalles de la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Ruby .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ejemplos de Amazon SES que utilizan el SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby con Amazon SES.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

Obtención del estado de una identidad

El siguiente ejemplo de código muestra cómo obtener el estado de una identidad de Amazon SES.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ses"  # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: "us-west-2")

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
```

```
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

- Para obtener más información sobre la API, consulta GetIdentityVerificationAttributesla Referencia AWS SDK for Ruby de la API.

Enumeración de identidades

En el siguiente ejemplo de código se muestra cómo enumerar identidades de Amazon SES.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ses"  # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: "us-west-2")

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
```

```
  })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

- Para obtener más información sobre la API, consulta ListIdentitiesla Referencia AWS SDK for Ruby de la API.

Enviar correos electrónicos

En el siguiente ejemplo de código se muestra cómo enviar un correo electrónico con Amazon SES.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ses"  # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = "sender@example.com"

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = "recipient@example.com"

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
```

```
subject = "Amazon SES test (AWS SDK for Ruby)"

# The HTML body of the email.
htmlbody =
  "<h1>Amazon SES test (AWS SDK for Ruby)</h1>"\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  "AWS SDK for Ruby</a>."

# The email body for recipients with non-HTML email clients.
textbody = "This email was sent with Amazon SES using the AWS SDK for Ruby."

# Specify the text encoding scheme.
encoding = "UTF-8"

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: "us-west-2")

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
```

```
    source: sender,
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
    )

  puts "Email sent to " + recipient

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"
end
```

- Para obtener más información sobre la API, consulta SendEmailla Referencia AWS SDK for Ruby de la API.

Verificación de una identidad de correo electrónico

El siguiente ejemplo de código muestra cómo verificar una identidad de correo electrónico con Amazon SES.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-ses"  # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = "recipient@example.com"

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: "us-west-2")

# Try to verify email address.
begin
```

```
  ses.verify_email_identity({
    email_address: recipient
  })

  puts "Email sent to " + recipient

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"
end
```

- Para obtener más información sobre la API, consulta VerifyEmailIdentityla Referencia AWS SDK for Ruby de la API.

Ejemplos de la API v2 de Amazon SES con SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK for Ruby API v2 de Amazon SES.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Enviar un correo electrónico

El siguiente ejemplo de código muestra cómo enviar un correo electrónico con la API v2 de Amazon SES.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sesv2"
require_relative "config" # Recipient and sender email addresses.

# Set up the SESv2 client.
client = Aws::SESV2::Client.new(region: AWS_REGION)

def send_email(client, sender_email, recipient_email)
  response = client.send_email(
    {
       from_email_address: sender_email,
       destination: {
         to_addresses: [recipient_email]
       },
       content: {
         simple: {
           subject: {
             data: "Test email subject"
           },
           body: {
             text: {
               data: "Test email body"
             }
           }
         }
       }
    }
  )
  puts "Email sent from #{SENDER_EMAIL} to #{RECIPIENT_EMAIL} with message ID:
 #{response.message_id}"
end

send_email(client, SENDER_EMAIL, RECIPIENT_EMAIL)
```

- Para obtener más información sobre la API, consulta SendEmailla Referencia AWS SDK for Ruby de la API.

Ejemplos de Amazon SNS con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Amazon SNS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones
- Ejemplos sin servidor

Acciones

Crear un tema

En el siguiente ejemplo de código se muestra cómo crear un tema de Amazon SNS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# This class demonstrates how to create an Amazon Simple Notification Service (SNS)
 topic.
```

```
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK for Ruby de la API.

Obtener la lista de los suscriptores de un tema

En el siguiente ejemplo de código se muestra cómo obtener la lista de suscriptores de un tema de Amazon SNS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# This class demonstrates how to list subscriptions to an Amazon Simple Notification
 Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = "SNS_TOPIC_ARN" # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)
```

```
  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para obtener más información sobre la API, consulta ListSubscriptionsla Referencia AWS SDK for Ruby de la API.

Enumeración de temas

En el siguiente ejemplo de código se muestra cómo enumerar temas de Amazon SNS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sns"  # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
rescue StandardError => e
  puts "Error while listing the topics: #{e.message}"
  end
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)
```

```
  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK for Ruby de la API.

Publicar en un tema

En el siguiente ejemplo de código se muestra cómo publicar mensajes en un tema de Amazon SNS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Service class for sending messages using Amazon Simple Notification Service (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end
```

```
  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message content

  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para obtener información acerca de la API, consulte Publish en la referencia de la API de AWS SDK for Ruby .

Crear atributos de temas

En el siguiente ejemplo de código se muestra cómo crear atributos de temas de Amazon SNS.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
```

```
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                   Sid: "__default_statement_ID",
                   Effect: "Allow",
                   Principal: { "AWS": "*" },
                   Action: ["SNS:Publish"],
                   Resource: topic_arn,
                   Condition: {
                     ArnEquals: {
                       "AWS:SourceArn": resource_arn
                     }
                   }
                 }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK for Ruby de la API.

Suscribir una dirección de correo electrónico a un tema

En el siguiente ejemplo de código se muestra cómo suscribir una dirección de correo electrónico a un tema de Amazon SNS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
```

```
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para obtener información acerca de la API, consulte Subscribe (Suscríbase) en la referencia de la API de AWS SDK for Ruby .

Ejemplos sin servidor

Invocar una función de Lambda desde un desencadenador de Amazon SNS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Uso de un evento de SNS con Lambda mediante Ruby.

```
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end
```

```
def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Ejemplos de Amazon SQS con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Amazon SQS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones
- Ejemplos sin servidor

Acciones

Cambiar la visibilidad del tiempo de espera de los mensajes

En el siguiente ejemplo de código se muestra cómo cambiar la visibilidad del tiempo de espera de un mensaje de Amazon SQS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sqs"  # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: "us-west-2")

begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  receive_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
      queue_url: queue_url,
      receipt_handle: message.receipt_handle,
      visibility_timeout: 30 # This message will not be visible for 30 seconds after
 first receipt.
    })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
  })

  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."
```

```
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{receive_queue_name}', as it
 does not exist."
end
```

- Para obtener más información sobre la API, consulta ChangeMessageVisibilityla Referencia AWS SDK for Ruby de la API.

Creación de una cola

En el siguiente ejemplo de código se muestra cómo crear una cola de Amazon SQS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# This code example demonstrates how to create a queue in Amazon Simple Queue
 Service (Amazon SQS).

require "aws-sdk-sqs"

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
```

```
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts "Queue created."
  else
    puts "Queue not created."
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta CreateQueuela Referencia AWS SDK for Ruby de la API.

Eliminar una cola

En el siguiente ejemplo de código se muestra cómo eliminar una cola de Amazon SQS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sqs"  # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: "us-west-2")
```

```
sqs.delete_queue(queue_url: URL)
```

- Para obtener más información sobre la API, consulta DeleteQueuela Referencia AWS SDK for Ruby de la API.

Mostrar colas

El siguiente ejemplo de código muestra cómo obtener una lista de colas de Amazon SQS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sqs"
require "aws-sdk-sts"

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-west-2'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
```

```
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: ["All"]
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end

rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Listing available queue URLs..."
  list_queue_urls(sqs_client)

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs." + region + ".amazonaws.com/" +
    sts_client.get_caller_identity.account + "/" + queue_name

  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end
```

- Para obtener más información sobre la API, consulta ListQueueslaReferencia AWS SDK for Ruby de la API.

Recibir mensajes de una cola

En el siguiente ejemplo de código se muestra cómo recibir mensajes de una cola de Amazon SQS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sqs"
require "aws-sdk-sts"

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     10
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)

  if max_number_of_messages > 10
    puts "Maximum number of messages to receive must be 10 or less. " \
      "Stopping program."
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )

  if response.messages.count.zero?
    puts "No messages to receive, or all messages have already " \
```

```
      "been previously received."
    return
  end

  response.messages.each do |message|
    puts "-" * 20
    puts "Message body: #{message.body}"
    puts "Message ID:   #{message.message_id}"
  end

rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs." + region + ".amazonaws.com/" +
    sts_client.get_caller_identity.account + "/" + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta ReceiveMessagela Referencia AWS SDK for Ruby de la API.

Enviar un lote de mensajes a una cola

En el siguiente ejemplo de código se muestra cómo enviar un lote de mensajes a una cola de Amazon SQS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sqs"
require "aws-sdk-sts"

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
```

```
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  entries = [
    {
      id: "Message1",
      message_body: "This is the first message."
    },
    {
      id: "Message2",
      message_body: "This is the second message."
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs." + region + ".amazonaws.com/" +
    sts_client.get_caller_identity.account + "/" + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts "Messages sent."
  else
    puts "Messages not sent."
  end
end
```

- Para obtener más información sobre la API, consulta SendMessageBatchla Referencia AWS SDK for Ruby de la API.

Enviar un mensaje a una cola

En el siguiente ejemplo de código se muestra cómo enviar un mensaje a una cola de Amazon SQS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require "aws-sdk-sqs"
require "aws-sdk-sts"

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
```

```
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  message_body = "This is my message."

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs." + region + ".amazonaws.com/" +
    sts_client.get_caller_identity.account + "/" + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending a message to the queue named '#{queue_name}'..."

  if message_sent?(sqs_client, queue_url, message_body)
    puts "Message sent."
  else
    puts "Message not sent."
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta SendMessagela Referencia AWS SDK for Ruby de la API.

Ejemplos sin servidor

Invocar una función de Lambda desde un desencadenador de Amazon SQS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de una cola de SQS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Uso de un evento de SQS con Lambda mediante Ruby.

```
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Amazon SQS.

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de una cola de SQS. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Notificación de los errores de los elementos del lote de SQS con Lambda mediante Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

AWS STS ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with AWS STS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Asumir un rol

El siguiente ejemplo de código muestra cómo asumir un rol con AWS STS.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
```

```
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- Para obtener más información sobre la API, consulta AssumeRolela Referencia AWS SDK for Ruby de la API.

WorkDocs Ejemplos de Amazon que utilizan SDK for Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby con Amazon WorkDocs.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

Describa el contenido de la carpeta raíz

El siguiente ejemplo de código muestra cómo describir el contenido de WorkDocs la carpeta raíz de Amazon.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # Retrieves the root folder for a user by email
 # @param users [Array<Types::User>] A list of users selected from API response
 # @param user_email [String] The email of the user.
 def get_user_folder(users, user_email)
   user = users.find { |user| user.email_address == user_email }
   if user
     user.root_folder_id
   else
     @logger.error "Could not get root folder for user with email address
#{user_email}"
     exit(1)
   end
 end

 # Describes the contents of a folder
 # @param [String] folder_id - The Id of the folder to describe.
 def describe_folder_contents(folder_id)
   resp = @client.describe_folder_contents({
                                             folder_id: folder_id, # required
                                             sort: "NAME", # accepts DATE, NAME
                                             order: "ASCENDING", # accepts
ASCENDING, DESCENDING
                                           })
   resp.documents.each do |doc|
     md = doc.latest_version_metadata
     @logger.info "Name:          #{md.name}"
     @logger.info "Size (bytes):  #{md.size}"
     @logger.info "Last modified: #{doc.modified_timestamp}"
     @logger.info "Doc ID:        #{doc.id}"
     @logger.info "Version ID:    #{md.id}"
     @logger.info ""
   end
 rescue Aws::WorkDocs::Errors::ServiceError => e
   @logger.error "Error listing folder contents: #{e.message}"
```

```
    exit(1)
  end
```

- Para obtener más información sobre la API, consulta DescribeRootFolderslla Referencia AWS SDK for Ruby de la API.

Describa a los usuarios

El siguiente ejemplo de código muestra cómo describir a WorkDocs los usuarios de Amazon.

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # Describes users within an organization
  # @param [String] org_id: The ID of the org.
  def describe_users(org_id)
    resp = @client.describe_users({
                                    organization_id: org_id,
                                    include: "ALL", # accepts ALL, ACTIVE_PENDING
                                    order: "ASCENDING", # accepts ASCENDING,
DESCENDING
                                    sort: "USER_NAME", # accepts USER_NAME,
FULL_NAME, STORAGE_LIMIT, USER_STATUS, STORAGE_USED
                                  })
    resp.users.each do |user|
      @logger.info "First name:  #{user.given_name}"
      @logger.info "Last name:   #{user.surname}"
      @logger.info "Email:       #{user.email_address}"
      @logger.info "Root folder: #{user.root_folder_id}"
      @logger.info ""
    end
    resp.users
  rescue Aws::WorkDocs::Errors::ServiceError => e
    @logger.error "AWS WorkDocs Service Error: #{e.message}"
    exit(1)
```

```
end
```

- Para obtener más información sobre la API, consulta DescribeUsersla Referencia AWS SDK for Ruby de la API.

Ejemplos de servicios combinados con SDK para Ruby

Las siguientes aplicaciones de ejemplo utilizan AWS SDK for Ruby para funcionar en varios Servicios de AWS.

Los ejemplos de servicios combinados apuntan a un nivel avanzado de experiencia para ayudarle a empezar a crear aplicaciones.

Ejemplos

- Crear una aplicación que analice los comentarios de los clientes y sintetice el audio

Crear una aplicación que analice los comentarios de los clientes y sintetice el audio

SDK para Ruby

Esta aplicación de ejemplo analiza y almacena las tarjetas de comentarios de los clientes. Concretamente, satisface la necesidad de un hotel ficticio en la ciudad de Nueva York. El hotel recibe comentarios de los huéspedes en varios idiomas en forma de tarjetas de comentarios físicas. Esos comentarios se cargan en la aplicación a través de un cliente web. Una vez cargada la imagen de una tarjeta de comentarios, se llevan a cabo los siguientes pasos:

- El texto se extrae de la imagen mediante Amazon Textract.
- Amazon Comprehend determina la opinión del texto extraído y su idioma.
- El texto extraído se traduce al inglés mediante Amazon Translate.
- Amazon Polly sintetiza un archivo de audio a partir del texto extraído.

La aplicación completa se puede implementar con el AWS CDK. Para ver el código fuente y las instrucciones de implementación, consulta el proyecto en GitHub.

Servicios utilizados en este ejemplo

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Seguridad para AWS SDK for Ruby

La seguridad en la nube de Amazon Web Services (AWS) es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes. La seguridad es una responsabilidad compartida entre usted AWS y usted. En el modelo de responsabilidad compartida, se habla de “seguridad de la nube” y “seguridad en la nube”:

Seguridad de la nube: AWS se encarga de proteger la infraestructura en la que se ejecutan todos los servicios que se ofrecen en la AWS nube y de proporcionarle servicios que pueda utilizar de forma segura. Nuestra responsabilidad en materia de seguridad es nuestra máxima prioridad AWS, y auditores externos comprueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los programas de AWS conformidad.

Seguridad en la nube: su responsabilidad viene determinada por el uso Servicio de AWS que utilice y por otros factores, como la confidencialidad de sus datos, los requisitos de su organización y las leyes y reglamentos aplicables.

Temas

Protección de datos en AWS SDK for Ruby

El modelo de responsabilidad AWS compartida modelo se aplica a la protección de datos en. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información

sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog AWS Shared Responsibility Model and GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-2 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja o Servicios de AWS utiliza la consola, la API o los SDK. AWS CLI AWS Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Identity and Access Management para AWS SDK for Ruby

AWS Identity and Access Management (IAM) es un servicio de Amazon Web Services (AWS) que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene

permisos) para utilizar recursos de Servicios de AWS. El IAM es un Servicio de AWS servicio que puede utilizar sin coste adicional.

Para poder acceder a AWS SDK for Ruby AWS, necesitas una AWS cuenta y AWS credenciales. Para aumentar la seguridad de su cuenta de AWS , le recomendamos que utilice un usuario de IAM a la hora de proporcionar credenciales de acceso en lugar de usar las credenciales de su cuenta de AWS .

Para obtener más información acerca de cómo trabajar con IAM, consulte IAM.

Para obtener información general sobre los usuarios de IAM y por qué son importantes para la seguridad de su cuenta, consulte Credenciales de seguridad de AWS en la Referencia general de Amazon Web Services.

AWS El SDK for Ruby sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información Servicio de AWS de seguridad, consulte la página Servicio de AWS de documentación de seguridad y Servicios de AWS consulte el alcance de las iniciativas de AWS cumplimiento implementadas por el programa de cumplimiento.

Validación de conformidad para AWS SDK for Ruby

AWS El SDK for Ruby sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información Servicio de AWS de seguridad, consulte la página Servicio de AWS de documentación de seguridad y Servicios de AWS consulte el alcance de las iniciativas de AWS cumplimiento implementadas por el programa de cumplimiento.

Auditores externos evalúan la seguridad y la conformidad de los servicios de Amazon Web Services (AWS) como parte de varios programas de AWS conformidad. Estos incluyen SOC, PCI, FedRAMP, HIPAA y otros. AWS proporciona una lista actualizada con frecuencia de los programas de cumplimiento específicos dentro del ámbito de aplicación Servicios de AWS en AWS Services in Scope by Compliance Program.

Los informes de auditoría de terceros están disponibles para que los descargue y los utilice AWS Artifact. Para obtener más información, consulta Descarga de informes en AWS Artifact.

Para obtener más información sobre los programas AWS de conformidad, consulte Programas AWS de conformidad.

Tu responsabilidad de cumplimiento al usar AWS SDK for Ruby para acceder a un Servicio de AWS está determinada por la confidencialidad de tus datos, los objetivos de cumplimiento de tu organización y las leyes y reglamentos aplicables. Si el uso de un Servicio de AWS está sujeto al cumplimiento de normas como HIPAA, PCI o FedRAMP, proporciona recursos que le ayudarán a: AWS

- Guías de inicio rápido sobre seguridad y conformidad: guías de implementación en las que se analizan las consideraciones arquitectónicas y se proporcionan los pasos necesarios para implementar entornos básicos centrados en la seguridad y el cumplimiento. AWS
- Documento técnico sobre cómo diseñar una arquitectura para la seguridad y el cumplimiento de la HIPAA: un documento técnico que describe cómo las empresas pueden utilizar para crear aplicaciones que cumplan con la HIPAA. AWS
- AWS Recursos de cumplimiento: una colección de libros de trabajo y guías que pueden aplicarse a su sector y ubicación.
- AWS Config: un servicio que evalúa en qué medida las configuraciones de sus recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub: una visión integral del estado de su seguridad AWS que le ayuda a comprobar su conformidad con los estándares y las mejores prácticas del sector de la seguridad.

Resiliencia para AWS SDK for Ruby

La infraestructura global de Amazon Web Services (AWS) se basa en Regiones de AWS zonas de disponibilidad.

Regiones de AWS proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia.

Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las zonas de disponibilidad Regiones de AWS y las zonas de disponibilidad, consulte Infraestructura global.AWS

AWS El SDK for Ruby sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información Servicio de AWS

de seguridad, consulte la página Servicio de AWS de documentación de seguridad y Servicios de AWS consulte el alcance de las iniciativas de AWS cumplimiento implementadas por el programa de cumplimiento.

Seguridad de infraestructura para AWS SDK for Ruby

AWS El SDK for Ruby sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información Servicio de AWS de seguridad, consulte la página Servicio de AWS de documentación de seguridad y Servicios de AWS consulte el alcance de las iniciativas de AWS cumplimiento implementadas por el programa de cumplimiento.

Para obtener información sobre los procesos AWS de seguridad, consulte el documento técnico AWS: Descripción general de los procesos de seguridad.

Imponer una versión mínima de TLS en el AWS SDK para Ruby

La comunicación entre el AWS SDK para Ruby y AWS está protegida mediante Secure Sockets Layer (SSL) o Transport Layer Security (TLS). Todas las versiones de SSL y las versiones de TLS anteriores a la 1.2 presentan vulnerabilidades que pueden comprometer la seguridad de la comunicación. AWS Por este motivo, debes asegurarte de usar el AWS SDK para Ruby con una versión de Ruby compatible con la versión 1.2 o posterior de TLS.

Ruby usa la biblioteca OpenSSL para proteger las conexiones HTTP. Las versiones compatibles de Ruby (1.9.3 y posteriores) instaladas mediante los administradores de paquetes del sistema (`yum`, `apt` y otros), un instalador oficial o administradores de Ruby (rbenv, RVM y otros) habitualmente incorporan OpenSSL 1.0.1 o posterior, que es compatible con TLS 1.2.

Cuando se usa con una versión compatible de Ruby con OpenSSL 1.0.1 o posterior, el SDK para AWS Ruby prefiere TLS 1.2 y usa la última versión de SSL o TLS compatible tanto con el cliente como con el servidor. Siempre es al menos para TLS 1.2. Servicios de AWS(El SDK utiliza la clase `Net::HTTP` de Ruby con `use_ssl=true`)

Comprobar la versión de OpenSSL

Para asegurarse de que su instalación de Ruby está usando OpenSSL 1.0.1 o posterior, introduzca el siguiente comando.

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

Otra forma de obtener la versión de OpenSSL es buscar directamente el ejecutable de `openssl`. Primero, localice el ejecutable adecuado mediante el siguiente comando.

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

La salida debe tener `--with-openssl-dir=/path/to/openssl` que indique la ubicación de la instalación de OpenSSL. Apunte esta ruta. Para comprobar la versión de OpenSSL, introduzca los siguientes comandos.

```
cd /path/to/openssl
bin/openssl version
```

Este último método puede no funcionar en todas las instalaciones de Ruby.

Actualizar la compatibilidad con TLS

Si la versión de OpenSSL utilizada por Ruby es inferior a 1.0.1, actualice la instalación de Ruby u OpenSSL mediante el administrador de paquetes de su sistema, el instalador de Ruby o el administrador de Ruby tal como se describe en la guía de instalación de Ruby. Si va a instalar Ruby desde el código fuente, instale primero la última versión de OpenSSL y pase `--with-openssl-dir=/path/to/upgraded/openssl` cuando ejecute `./configure`.

Migración de Amazon S3 Encryption Client

En este tema se muestra cómo migrar las aplicaciones de la versión 1 (V1) del cliente de cifrado Amazon Simple Storage Service (Amazon S3) a la versión 2 (V2) y cómo garantizar la disponibilidad de las aplicaciones durante todo el proceso de migración.

Información general sobre la migración

Esta migración se produce en dos fases:

1. Actualice los clientes existentes para leer nuevos formatos. Primero, implementa una versión actualizada del AWS SDK for Ruby en tu aplicación. Así, permitirá a los clientes de cifrado de la versión V1 descifrar los objetos escritos por los nuevos clientes de la versión V2. Si tu aplicación usa varios AWS SDK, debes actualizar cada SDK por separado.

2. Migue los clientes de cifrado y descifrado a la versión V2. Una vez que todos sus clientes de cifrado de la versión 1 puedan leer los nuevos formatos, puede migrar los clientes de cifrado y descifrado existentes a sus respectivas versiones de la versión 2.

Actualizar los clientes existentes para leer nuevos formatos

El cliente de cifrado de la versión V2 utiliza algoritmos de cifrado que las versiones anteriores del cliente no admiten. El primer paso de la migración consiste en actualizar los clientes de descifrado de la versión V1 a la versión más reciente del SDK. Tras completar este paso, los clientes de la versión V1 de su aplicación podrán descifrar los objetos cifrados por los clientes de cifrado de la versión V2. Consulta los detalles de cada versión principal del AWS SDK for Ruby a continuación.

Actualización AWS del SDK para Ruby versión 3

La versión 3 es la última versión del AWS SDK para Ruby. Para completar la migración, debe utilizar la versión 1.76.0 o una posterior de la gema `aws-sdk-s3`.

Instalación desde la línea de comandos

Para los proyectos que instalan la gema `aws-sdk-s3`, utilice la opción de versión para comprobar que está instalada la versión mínima de 1.76.0.

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

Uso de archivos Gemfile

Establezca la versión mínima de la gema `aws-sdk-s3` en 1.76.0 para los proyectos que utilizan un archivo Gemfile para gestionar las dependencias. Por ejemplo:

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. Modifique su archivo Gemfile.
2. Ejecute `bundle update aws-sdk-s3`. Para comprobar su versión, ejecute `bundle info aws-sdk-s3`.

Actualización del AWS SDK para Ruby versión 2

La versión 2 del AWS SDK for Ruby entrará en modo de mantenimiento el 21 de noviembre de 2021. Para completar la migración, debe utilizar la versión 2.11.562 o una posterior de la gema aws-sdk.

Instalación desde la línea de comandos

Para los proyectos que instalan la gema `aws-sdk`, utilice la opción de versión para comprobar que está instalada la versión mínima de 2.11.562.

```
gem install aws-sdk -v '>= 2.11.562'
```

Uso de archivos Gemfile

Establezca la versión mínima de la gema `aws-sdk` en 2.11.562 para los proyectos que utilizan un archivo Gemfile para gestionar las dependencias. Por ejemplo:

```
gem 'aws-sdk', '>= 2.11.562'
```

1. Modifique su archivo Gemfile. Si tiene un archivo Gemfile.lock, elimínelo o actualícelo.
2. Ejecute `bundle update aws-sdk`. Para comprobar su versión, ejecute `bundle info aws-sdk`.

Migrar clientes de cifrado y descifrado a la versión V2

Después de actualizar sus clientes para leer los nuevos formatos de cifrado, puede actualizar sus aplicaciones a la versión V2 de los clientes de cifrado y descifrado. En los siguientes pasos, se muestra cómo migrar correctamente el código de la versión V1 a la V2.

Antes de actualizar el código para usar el cliente de cifrado V2, siga los pasos anteriores y utilice la gema `aws-sdk-s3` en la versión 2.11.562 o posterior.

> **Note**
>
> Al descifrar con AES-GCM, lea todo el objeto hasta el final antes de empezar a utilizar los datos descifrados. Esto se hacer para verificar que el objeto no se ha modificado desde que se cifró.

Configuración de clientes de cifrado de la versión V2

El EncryptionV2::Client requiere una configuración adicional. Para obtener información de configuración detallada, consulte la documentación de EncryptionV2::Client o los ejemplos que se proporcionan más adelante en este tema.

1. El método de encapsulación de clave y el algoritmo de cifrado del contenido deben especificarse al construir el cliente. Al crear un `EncryptionV2::Client` nuevo, debe proporcionar valores para `key_wrap_schema` y `content_encryption_schema`.

`key_wrap_schema`- Si lo está utilizando AWS KMS, debe estar configurado en`:kms_context`. Si utiliza una clave simétrica (AES), esta se debe establecer en `:aes_gcm`. Si utiliza una clave simétrica (AES), esta se debe establecer en `:rsa_oaep_sha1`.

`content_encryption_schema` - Este se debe establecer en aes_gcm_no_padding.

2. security_profile debe especificarse en la construcción del cliente. Al crear un `EncryptionV2::Client` nuevo, debe proporcionar un valor para `security_profile`. El parámetro security_profile determina la compatibilidad para los objetos de lectura escritos con la versión V1 de `Encryption::Client` anterior. Hay dos valores: :v2 y :v2_and_legacy. Para admitir la migración, establezca el `security_profile` en :v2_and_legacy. Use :v2 solo para el desarrollo de nuevas aplicaciones.

3. AWS KMS key se aplica de forma predeterminada. En la `kms_key_id` versión `1Encryption::Client`, no se proporcionó al AWS KMS `Decrypt call`. AWS KMS puede obtener esta información de los metadatos y añadirla al blob simétrico de texto cifrado. En la versión 2, e`EncryptionV2: :Client`, el kms_key_id se pasa a la llamada Decrypt y la llamada falla si no coincide con la clave utilizada para AWS KMS cifrar el objeto. Si anteriormente no estableció un `kms_key_id` específico para su código, establezca `kms_key_id: :kms_allow_decrypt_with_any_cmk` en la creación del cliente o establezca `kms_allow_decrypt_with_any_cmk: true` en llamadas de `get_object`.

Ejemplo: Uso de una clave simétrica (AES)

Antes de la migración

```
client = Aws::S3::Encryption::Client.new(encryption_key: aes_key)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Después de la migración

```
client = Aws::S3::EncryptionV2::Client.new(
  encryption_key: rsa_key,
  key_wrap_schema: :rsa_oaep_sha1, # the key_wrap_schema must be rsa_oaep_sha1 for
 asymmetric keys
```

```
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
 )
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key) # No changes
```

AWS KMS Ejemplo: usar con kms_key_id

Antes de la migración

```
client = Aws::S3::Encryption::Client.new(kms_key_id: kms_key_id)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Después de la migración

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key) # No change
```

Ejemplo: usar sin kms_key_id AWS KMS

Antes de la migración

```
client = Aws::S3::Encryption::Client.new(kms_key_id: kms_key_id)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Después de la migración

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
```

```
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key, kms_allow_decrypt_with_any_cmk:
 true) # To allow decrypting with any cmk
```

Alternativa posterior a la migración

Si solo lee y descifra (nunca escribe ni cifra) objetos con el cliente de cifrado de S2, utilice este código.

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: :kms_allow_decrypt_with_any_cmk, # set kms_key_id to allow all get_object
 requests to use any cmk
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
resp = client.get_object(bucket: bucket, key: key) # No change
```

Historial de revisión

En la siguiente tabla se describen cambios importantes en esta guía. Para obtener notificaciones sobre las actualizaciones de esta documentación, puede suscribirse a una fuente RSS.

Cambio	Descripción	Fecha
Índice	Se ha actualizado el índice para que los ejemplos de código sean más accesibles.	1 de junio de 2023
Actualizaciones de las prácticas recomendadas de IAM	Se ha actualizado la guía para implementar las prácticas recomendadas de IAM. Para obtener más información, consulte Prácticas recomendadas de seguridad en IAM. Se ha actualizado el tutorial de introducción.	8 de mayo de 2023
Actualizaciones generales	Se ha actualizado la página de bienvenida de los recursos externos pertinentes. También se ha actualizado la versión mínima requerida de Ruby para la versión 2.3. Se han actualizado las secciones sobre AWS Key Management Service para reflejar las actualizaciones terminológicas. Se ha actualizado la información de uso de la utilidad REPL para favorecer la claridad.	8 de agosto de 2022
Corrección de enlaces que no funcionaban	Se han corregido enlaces que no funcionaban. Se ha	3 de agosto de 2022

	eliminado la página redundante de consejos y trucos; se ha redirigido al contenido de ejemplo de Amazon EC2. Se han añadido listas de ejemplos de código disponibles en GitHub al repositorio de ejemplos de código.	
Obtener información acerca de todos los grupos de seguridad de Amazon RDS	Se han agregado notas sobre la retirada de EC2-Classic.	26 de julio de 2022
SDK Metrics	Se ha eliminado la información sobre la activación de SDK Metrics for Enterprise Support, que había quedado obsoleta.	28 de enero de 2022

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.